

# Large Language Models (LLMs)

Introduction to Large Language Models (LLM)

**Zeham Management Technologies BootCamp**

**by SDAIA**

**September 16<sup>th</sup>, 2024**



**SDAIA**  
الهيئة السعودية للبيانات  
والذكاء الاصطناعي  
Saudi Data & AI Authority



## Agenda

**Power of Generative AI**

**What is a Language Model**

**Large Language Models**

**Perplexity Example**

**Revisiting Basic Terminologies**

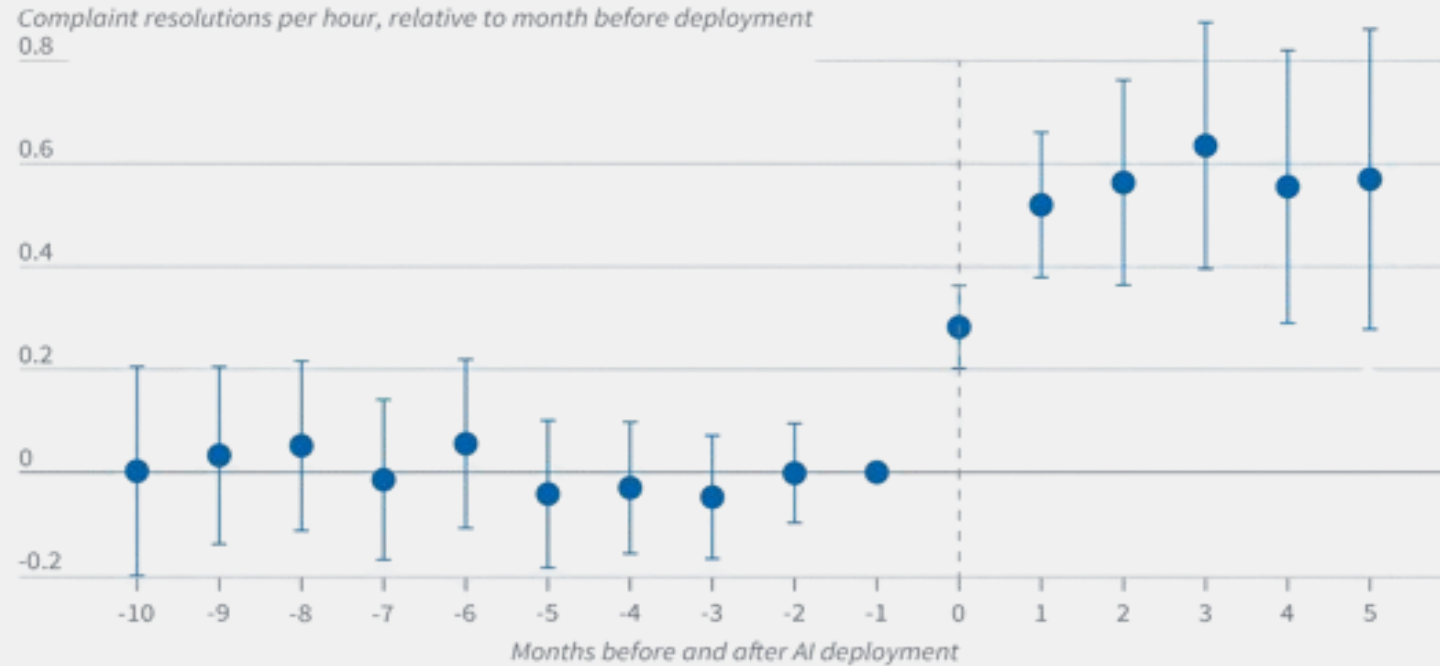
**Prompt Engineering**



# Power of Generative AI



## AI Assistance and Customer Complaint Resolutions



*Thin bars represent 95% confidence intervals*

*Source: Researchers' calculations using data from customer support agents provided by a Fortune 500 enterprise software company*





[Source](#)



من سار على الدرب .....



# What is a language model?

A language model refers to a type of model specifically designed to generate human-like text or predict the probability of a sequence of words. Language models learn patterns and statistics from large amounts of text data, enabling them to generate sensible and contextually appropriate sentences.



# ▶ What is a language model?

The cat	eats	0.3
	stays	0.2
	sat	0.5





# ▶ What is a language model?

The cat sat


at	0.1
in	0.1
on	0.8

A diagram illustrating a language model's output. The phrase 'The cat sat' is followed by three possible words: 'at', 'in', and 'on'. Each word is associated with a probability: 'at' is 0.1, 'in' is 0.1, and 'on' is 0.8. Three orange arrows point from 'sat' to each word. The 'on' row is enclosed in a green dashed box, indicating it is the most likely next word.

# ▶ What is a language model?

The cat sat on

floor	0.1
the	0.9
zoo	0.0



# ▶ What is a language model?

The cat sat on the

house	0.1
mat	0.7
TV	0.2

Three orange arrows point from the text 'The cat sat on the' to the words 'house', 'mat', and 'TV' in the table. A green dotted box encloses the 'mat' row in the table.



# What is a language model?

The cat sat on the mat





# Large Language Models?

**Extensive Training Data:** Large language models are trained on vast amounts of text data, often comprising billions or even trillions of words. This extensive training data helps the models learn patterns, grammar, context, and a wide range of language nuances, enabling them to generate coherent and contextually appropriate responses.





# Large Language Models?

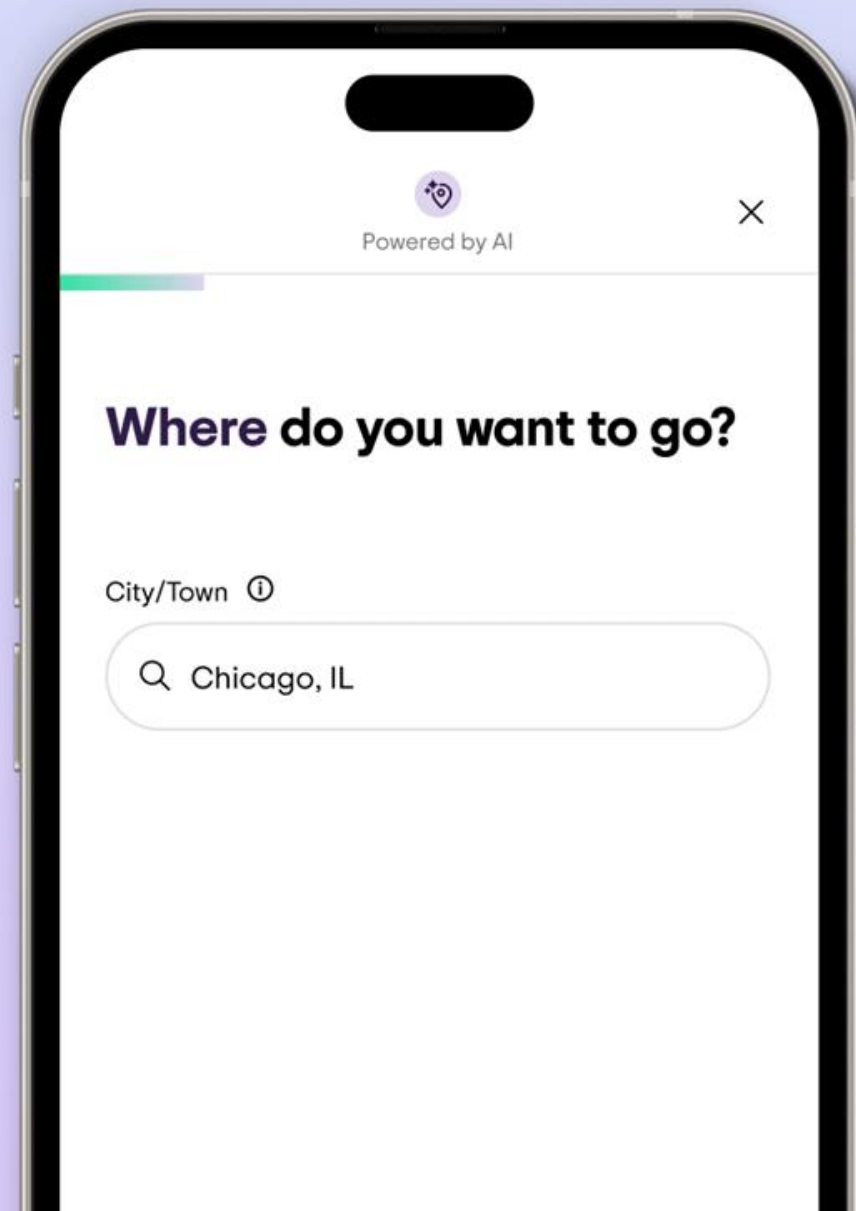
**Complex Architectures:** Large language models employ complex architectures, such as transformer networks, that contain numerous layers and millions or even billions of parameters. These architectures enable the models to capture intricate language structures, understand semantics, and generate high-quality text by leveraging the vast amount of training data they have been exposed to. The large number of parameters allows the models to learn fine-grained details and provide nuanced responses.



# Perplexity Example

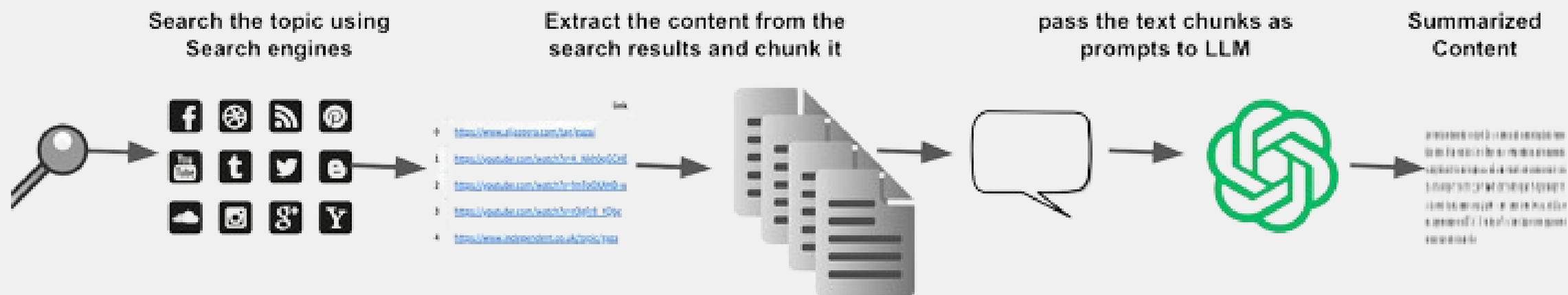


# Kick-start your travel planning



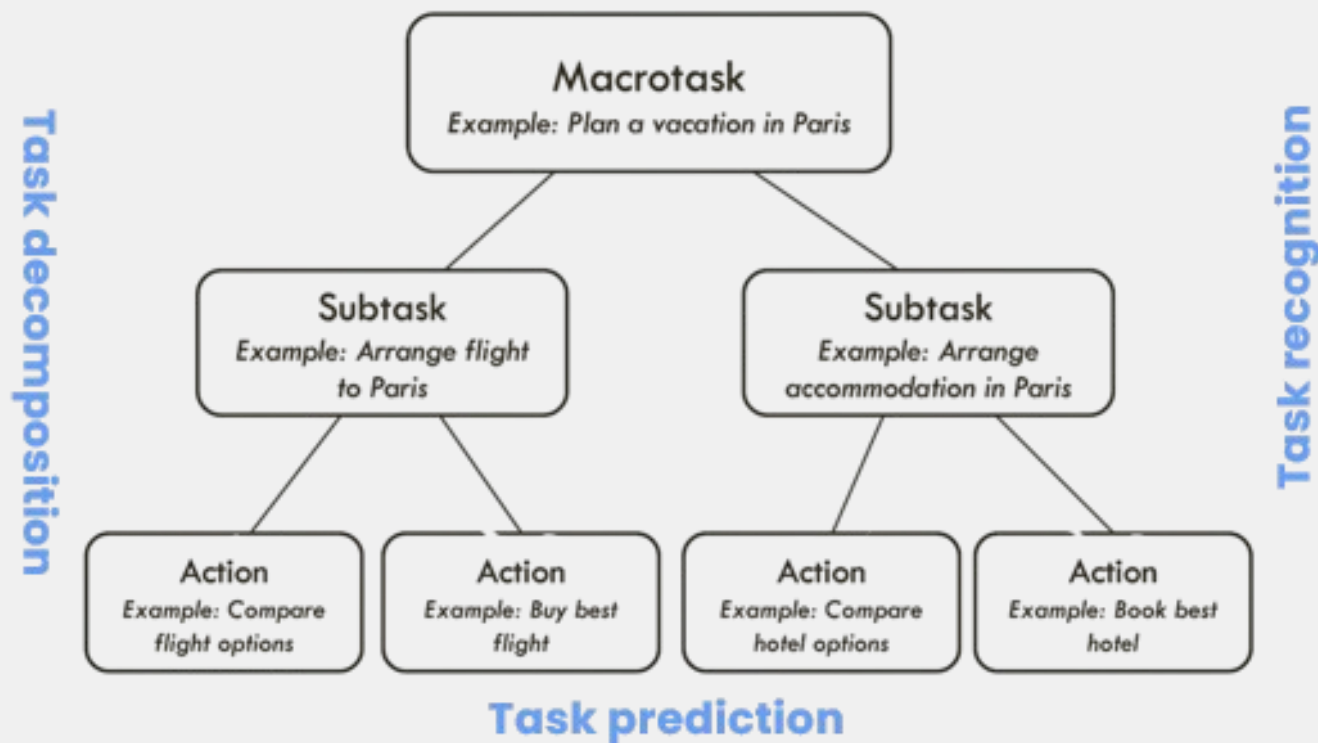


# Perplexity Example



# Perplexity Example

Dividing a complex search into subtasks, to then retrieve the results from them, and combine them into **one answer, by searching the internet is quite a hard task**





# Perplexity Example

Complex search is difficult, multi-faceted and requires deeper engagement

- There are 10 billion search queries a day, **an estimated half of them go unanswered.**
- That's because, **people are using search to do things it wasn't originally designed to do.**
- It's great for finding a website, but **for more complex questions or tasks, too often it falls short.**
- Complex search tasks involve **multiple queries, multiple sessions or requires deep engagement with search**



**Let's revisit some basic terminologies**



# Machine Learning Models

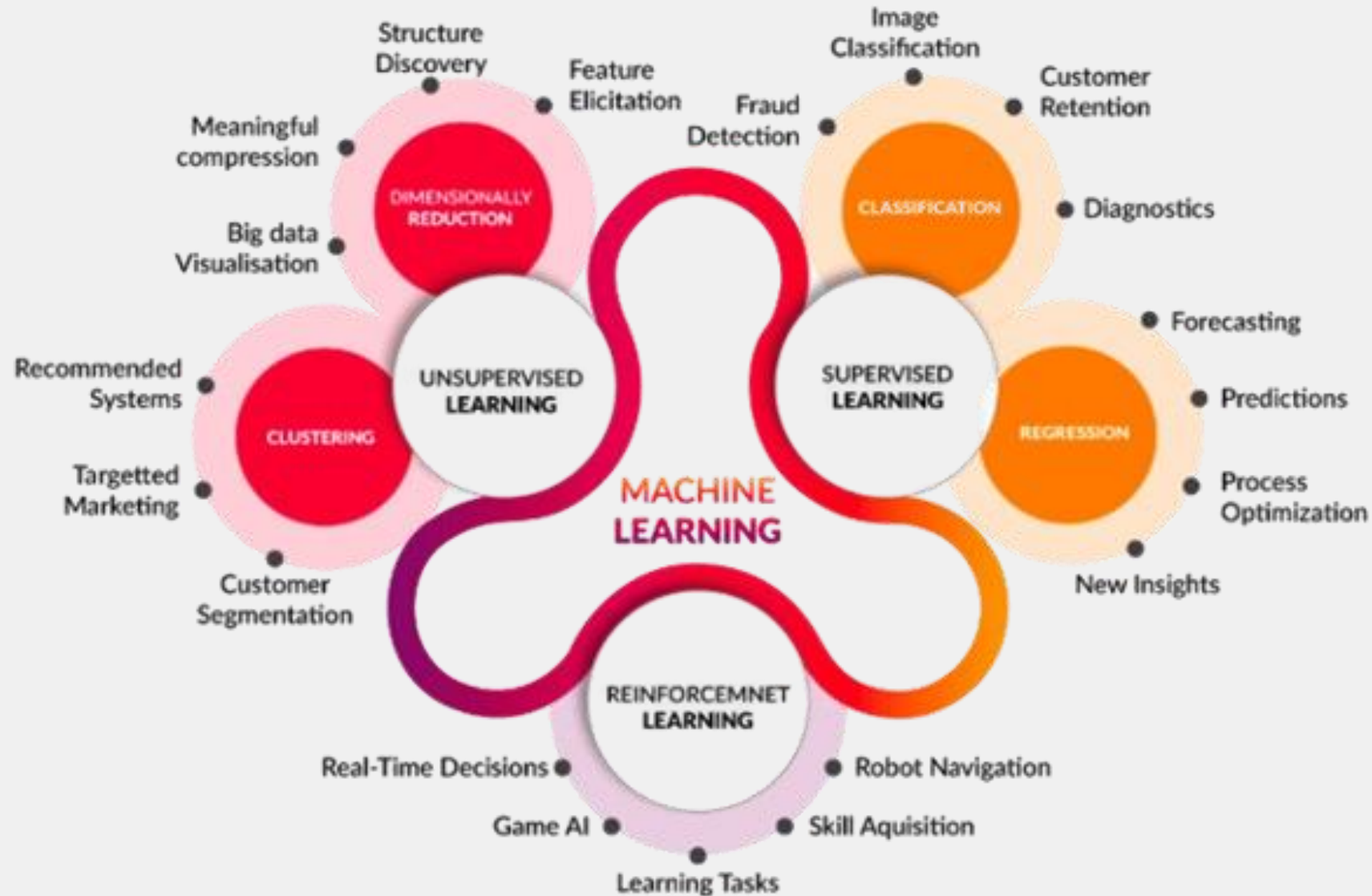
Machine learning is a discipline that focuses on creating algorithms and software capable of “learning” from the information we provide, and effectively perform specific types of tasks, as a result of this training.

Some examples:

- Netflix Recommendation Models
- Sales Prediction
- Self-Driving Cars



# Machine Learning Models





# Generative AI

Generative artificial intelligence (generative AI) is a type of AI that can create new content and ideas, including conversations, stories, images, videos, and music.

AI technologies attempt to mimic human intelligence in nontraditional computing tasks like image recognition, natural language processing (NLP), and translation.





# Generative AI market in focus

## Use case

## Applications

### Text

#### Marketing



#### Creative writing



#### Sales

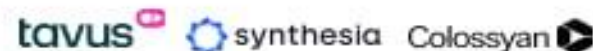


### Video

#### Generation & editing



#### Personalized video

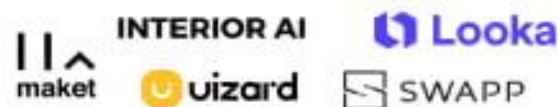


### Image

#### Generation & editing



#### Design



### Code

#### Code generation



#### App building



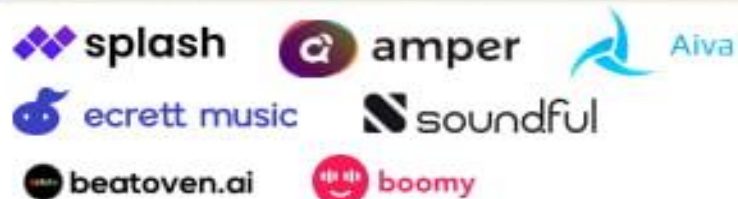


## Use case

## Applications

### Music

#### Generation & editing



#### Music customization



### Speech

#### Text-speech / speech-text

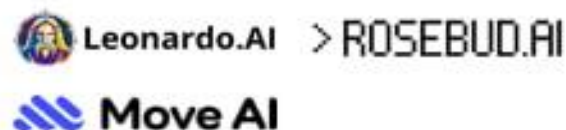


### 3D

#### Art



#### Gaming



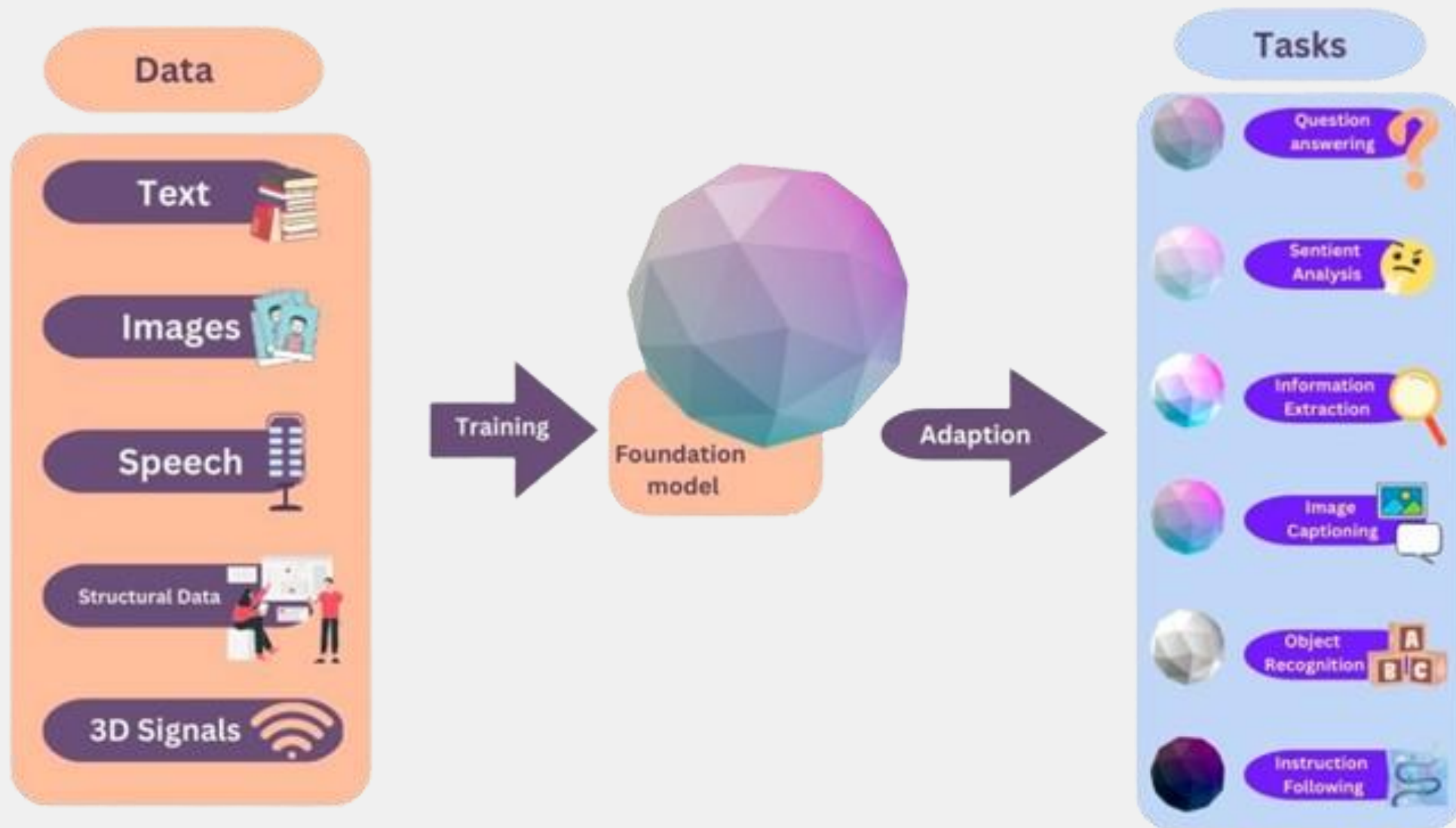


# Foundational Models

Foundation Model : A foundation model (also called base model) is a large machine learning (ML) model trained on a vast quantity of data at scale (often by self-supervised learning or semi-supervised learning), such that it can be adapted to a wide range of downstream tasks.

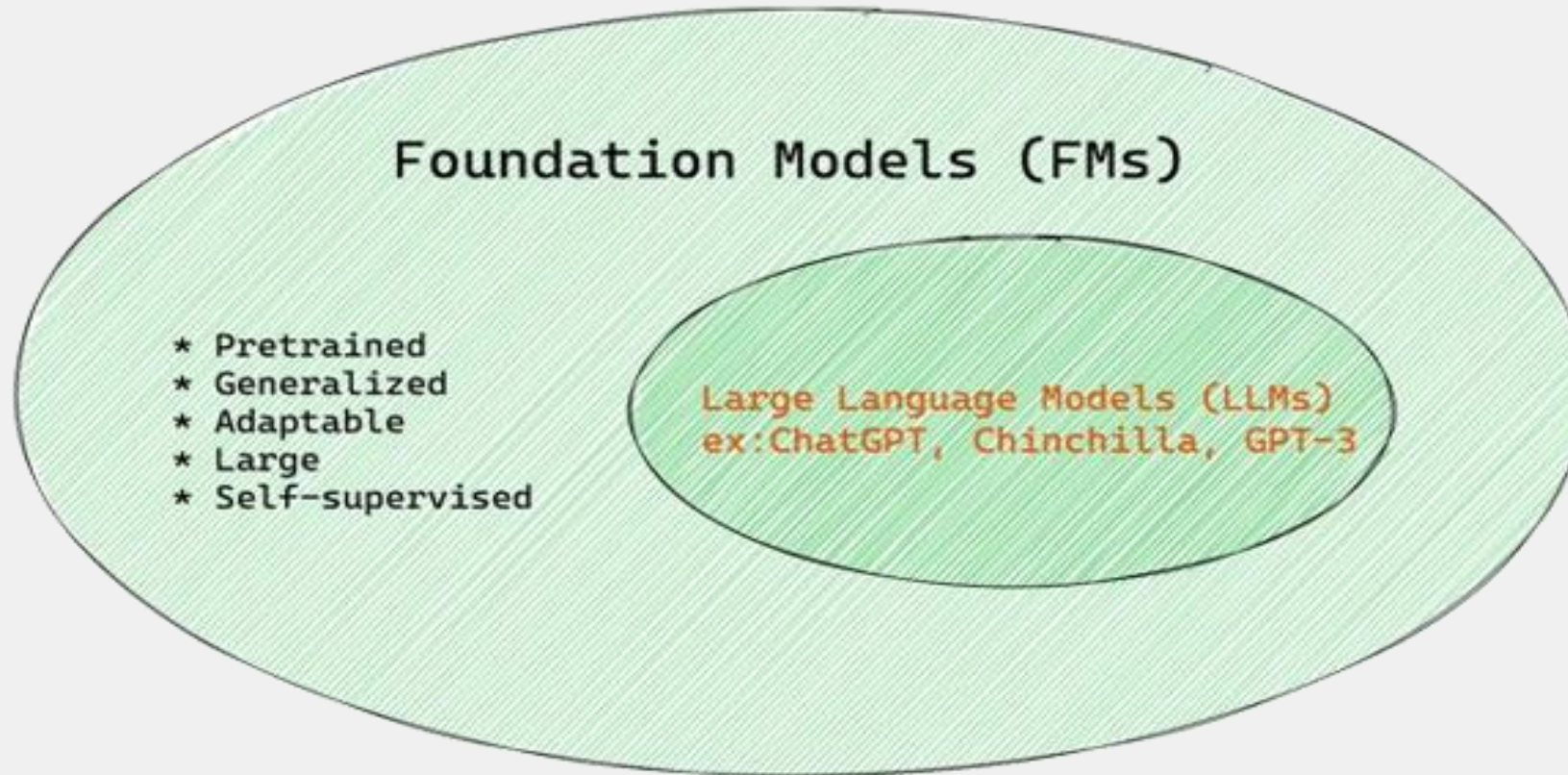


# Generative AI





# Generative AI



FMs are models trained on broad data (using self-supervision at scale) that can be adapted to wide range of downstream tasks.

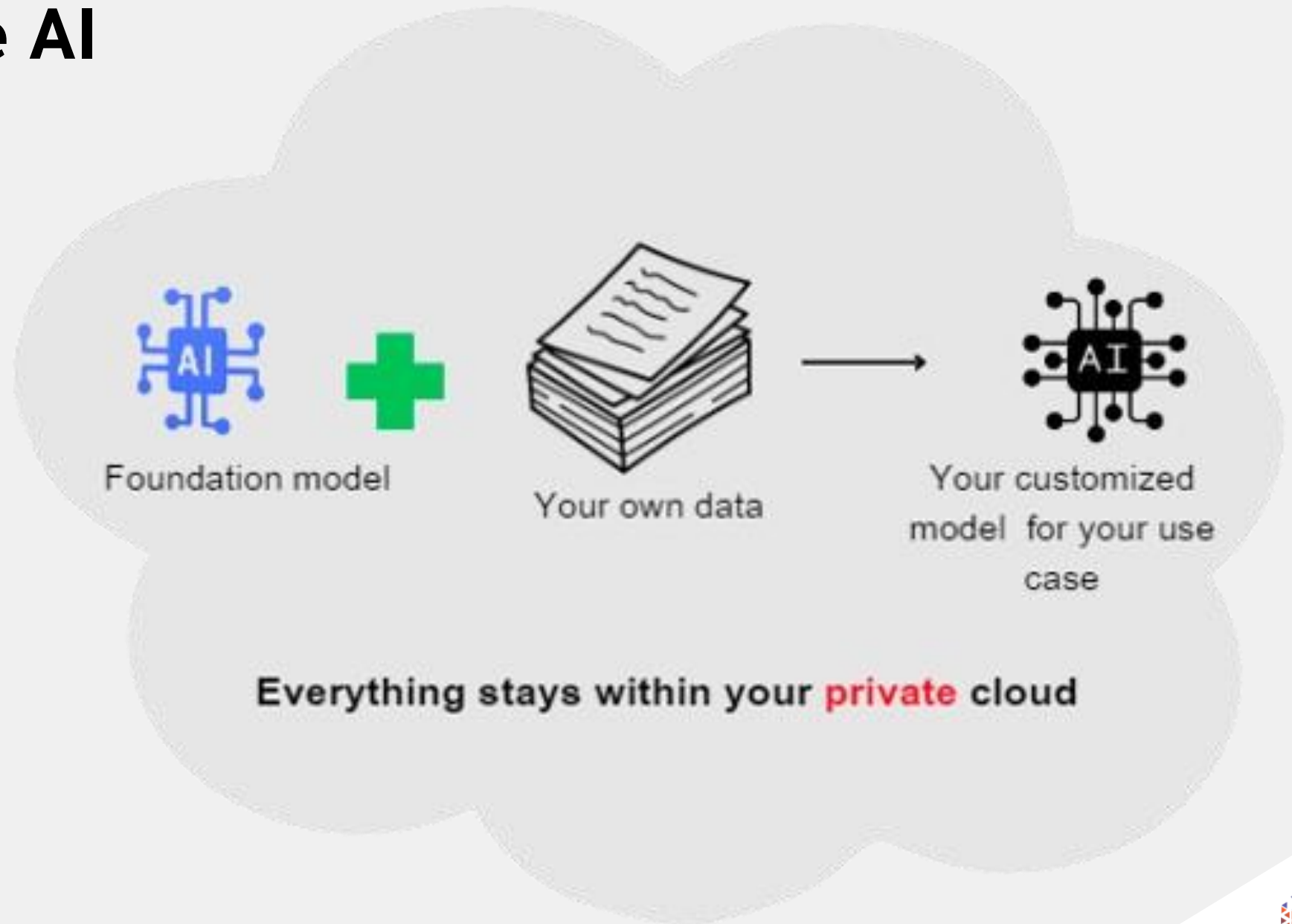
<https://hai.stanford.edu/news/reflections-foundation-models>





# Generative AI

Fine-tuning





# Foundational Models VS LLM

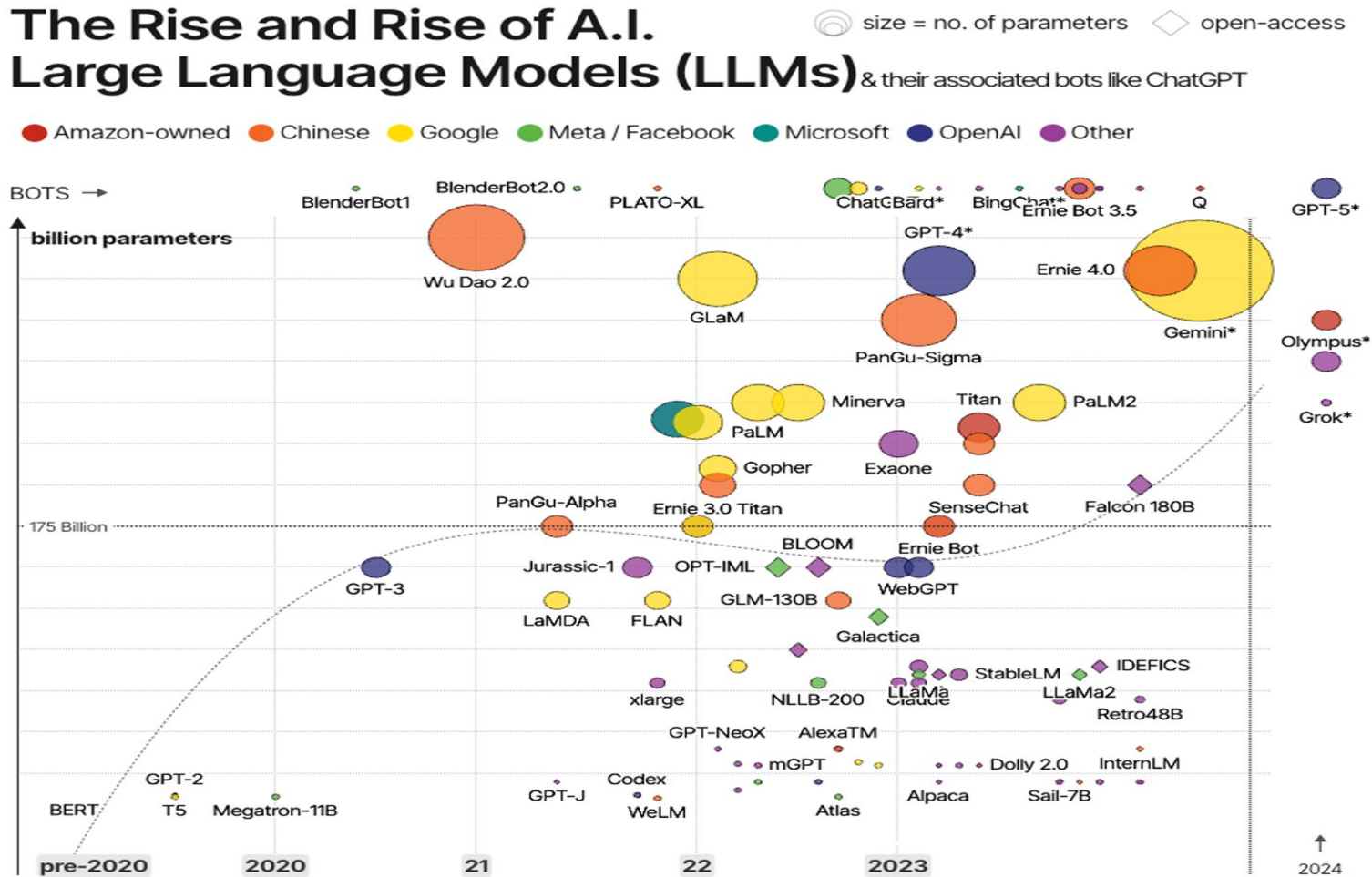
Large Language Models (LLMs) differ from Foundational Models in their scope of language understanding.

LLMs are specifically focused on language-related systems, while Foundational Models are attempting to stake out a broader function-based concept, which could accommodate new types of systems in the future





# The Rise and Rise of A.I. Large Language Models (LLMs) & their associated bots like ChatGPT



David McCandless, Tom Evans, Paul Barton  
Information is Beautiful // UPDATED 6th Dec 23

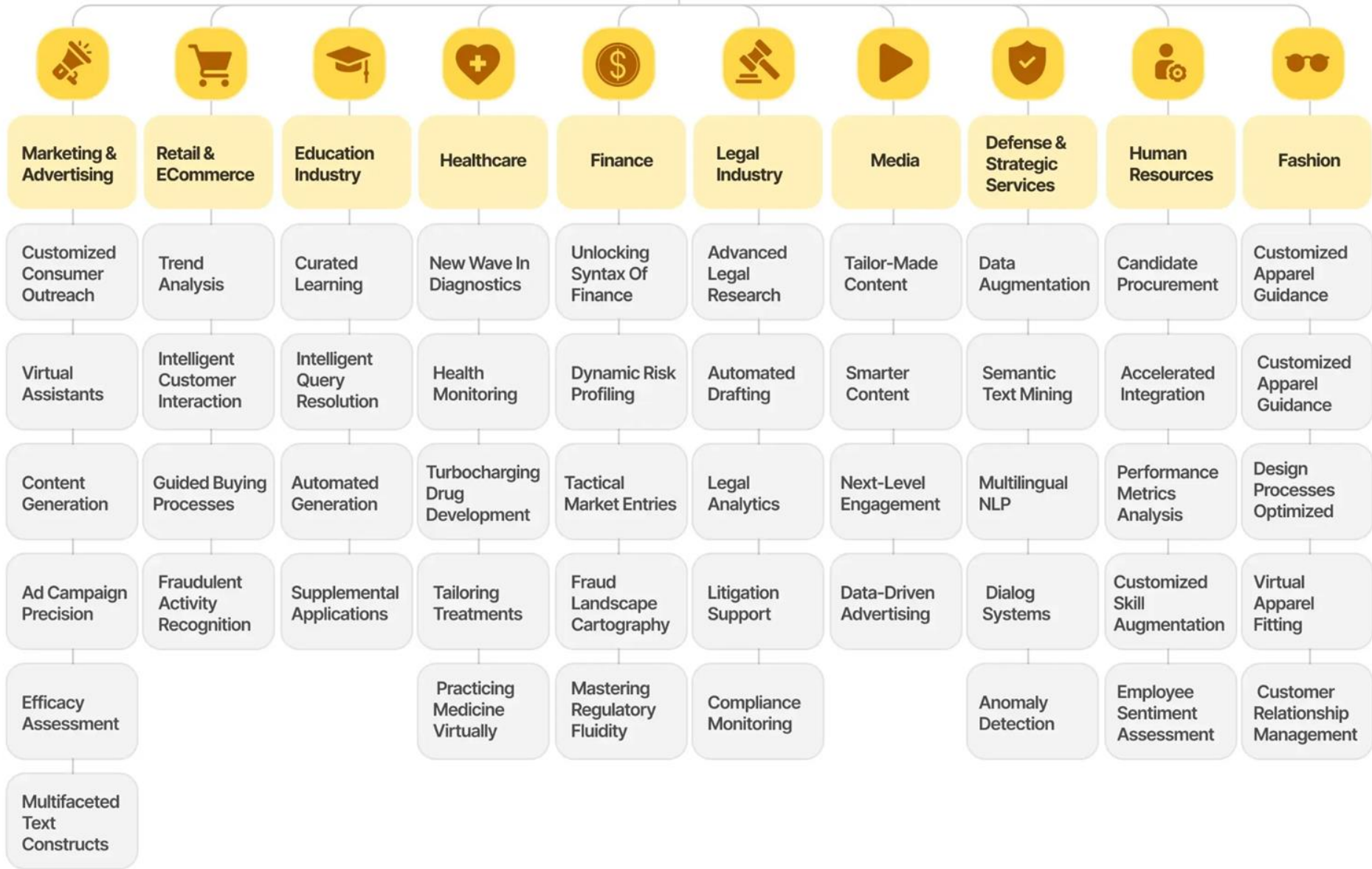
source: news reports, [LifeArchitect.ai](https://lifeaiarchitect.ai)  
\* = parameters undisclosed // see [the data](#)



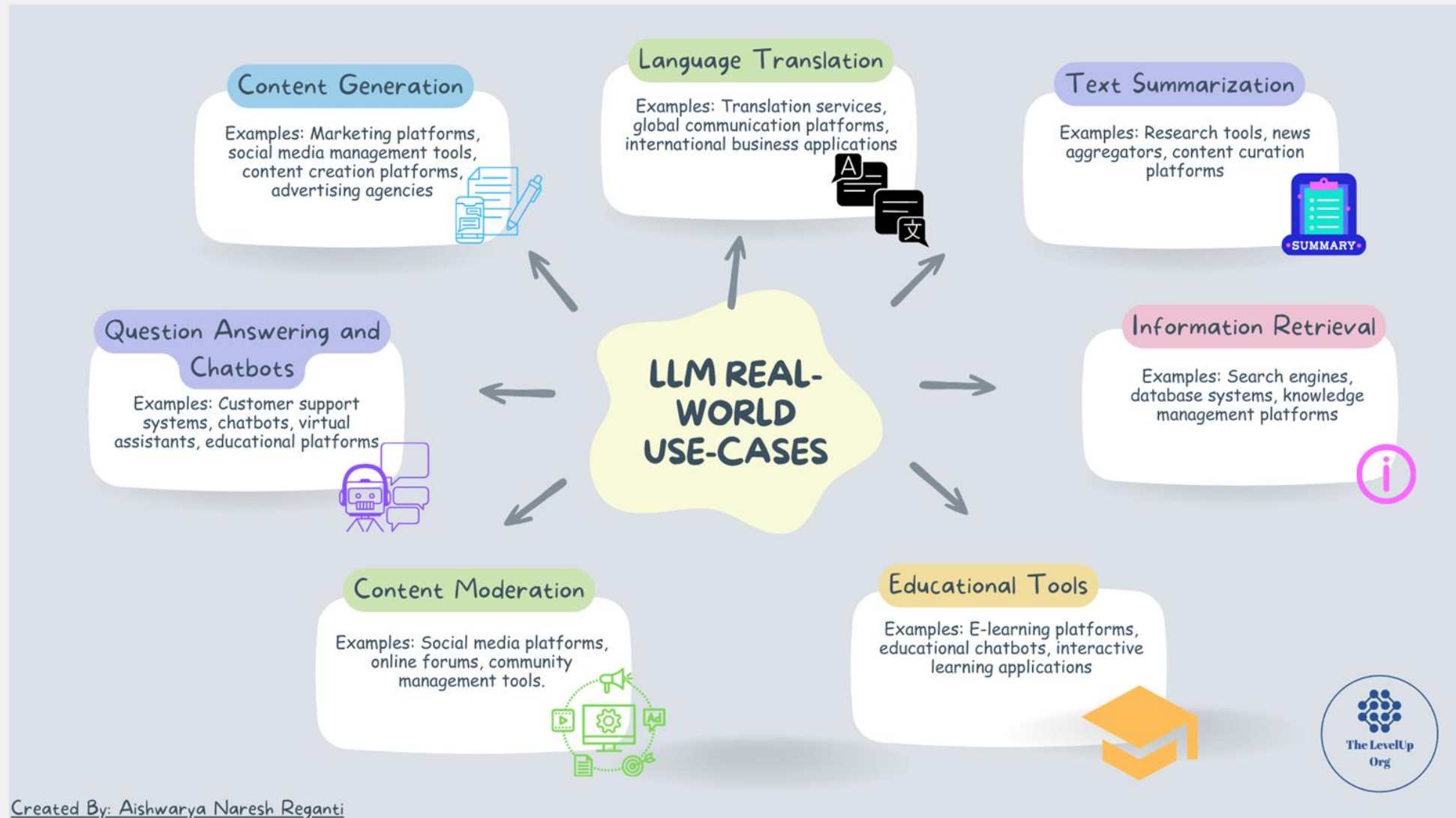




# LLM APPLICATIONS IN TOP INDUSTRIES







# Prompt Engineering

# What is Prompt Engineering ?

- Prompt engineering is the practice of designing and refining prompts—questions or instructions—to elicit specific responses from AI models.
- Think of it as the interface between human intent and machine output.





# Importance of Prompt Engineering ?

- **Reduces Ambiguity:** Clarifies user intent, leading to more precise and relevant outputs from AI models.
- **Improves User Experience:** Provides users with more accurate and helpful responses, improving satisfaction and engagement.



# Why Prompt Engineering works ?

- LLMs have been trained with extensive data and human annotated prompts.
- Hence when we use prompts to define role, details and intent, it yields better results
- Minimizes misunderstandings and errors by clearly defining the input parameters and expected output.

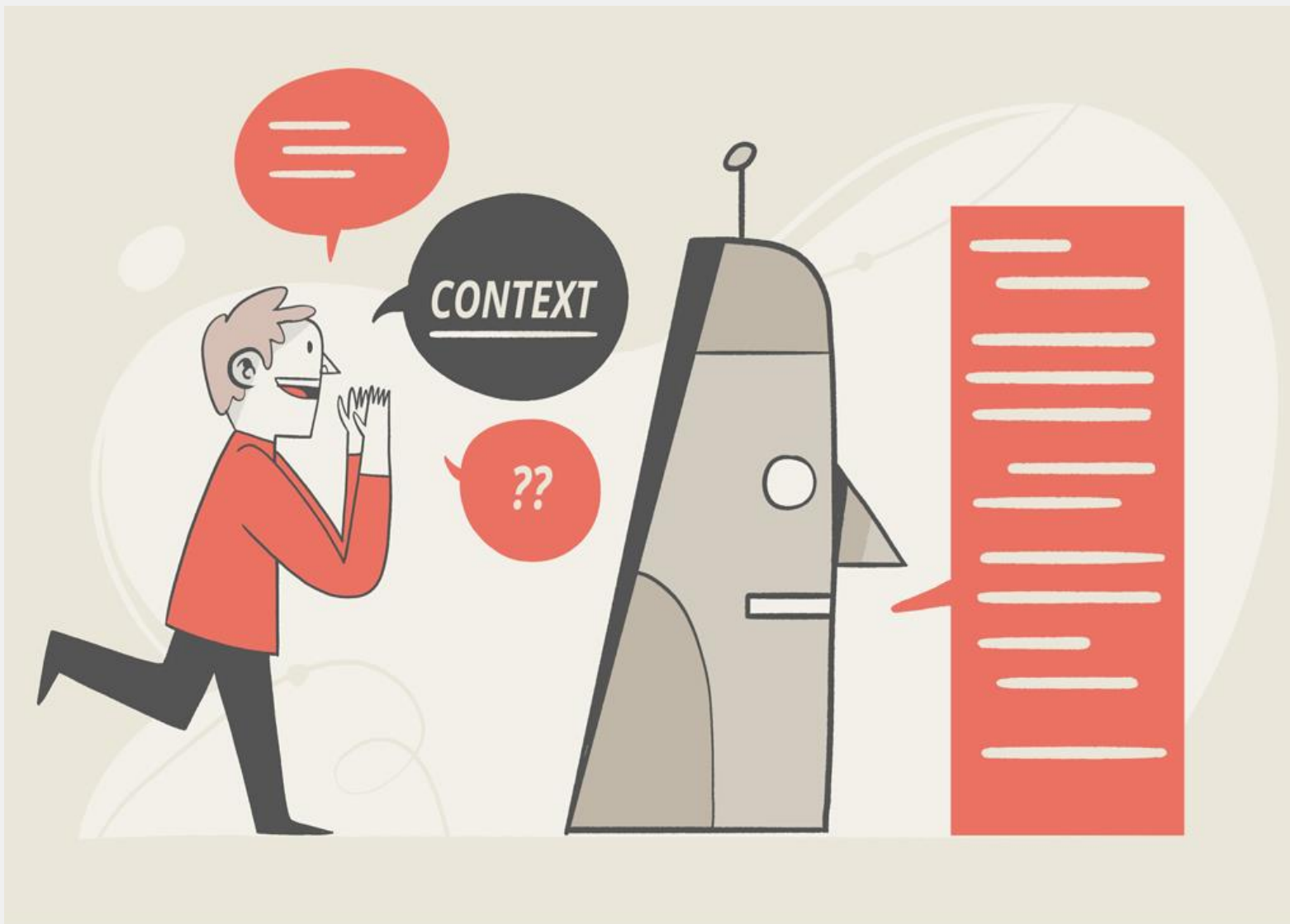


# Domain Adapted LLMs

There is a limitation on traditional models: they lack the specialization needed for specific industries or tasks. This is the reason why, often times, we require a **domain adapted LLM** for our *LLM related tasks*.

These LLMs are tailored to user specific needs and have improved interactions in the domain they are set to work in





Source: [Link](#)



# The Science of Prompt Engineering

Prompt engineering is the art of crafting precise prompts to guide the AI into delivering a desired output.

## Importance in Domain Adaptation:

- **Targeted Responses** → Ensures responses are not only accurate but also contextually relevant.
- **Leverage Domain Knowledge** → Incorporates specific terminology and workflows unique to the domain.

## Different prompting Strategies:

- **Instruction-based** → Direct and explicit instructions for tasks.
- **Contextual Clues** → Incorporating specific context to guide the AI's focus.





# The Science of Prompt Engineering

Prompt engineering is the art of crafting precise prompts to guide the AI into delivering a desired output.

## Importance in Domain Adaptation:

- **Targeted Responses** → Ensures responses are not only accurate but also contextually relevant.
- **Leverage Domain Knowledge** → Incorporates specific terminology and workflows unique to the domain.

## Different prompting Strategies:

- **Instruction-based** → Direct and explicit instructions for tasks.
- **Contextual Clues** → Incorporating specific context to guide the AI's focus.



# ▶ What are some more Advanced Prompt Engineering Techniques?

- **Zero-shot Prompting** → Utilizing generic prompts without prior examples.
- **Few-shot Prompting** → Providing a few examples to set a response pattern.
- **Fine-tuning with Examples** → Intensive example-based training to refine model understanding.

**Impact of Prompt Engineering** → Enhanced model performance, reliability, and domain-specific accuracy.



# ▶ What are some more Advanced Prompt Engineering Techniques?

- **Zero-shot Prompting** → Utilizing generic prompts without prior examples.
- **Few-shot Prompting** → Providing a few examples to set a response pattern.
- **Fine-tuning with Examples** → Intensive example-based training to refine model understanding.

**Impact of Prompt Engineering** → Enhanced model performance, reliability, and domain-specific accuracy.



# ▶ What are some more Advanced Prompt Engineering Techniques?

- **Zero-shot Prompting** → Utilizing generic prompts without prior examples.
- **Few-shot Prompting** → Providing a few examples to set a response pattern.
- **Fine-tuning with Examples** → Intensive example-based training to refine model understanding.

**Impact of Prompt Engineering** → Enhanced model performance, reliability, and domain-specific accuracy.



# ▶ What are some more Advanced Prompt Engineering Techniques?

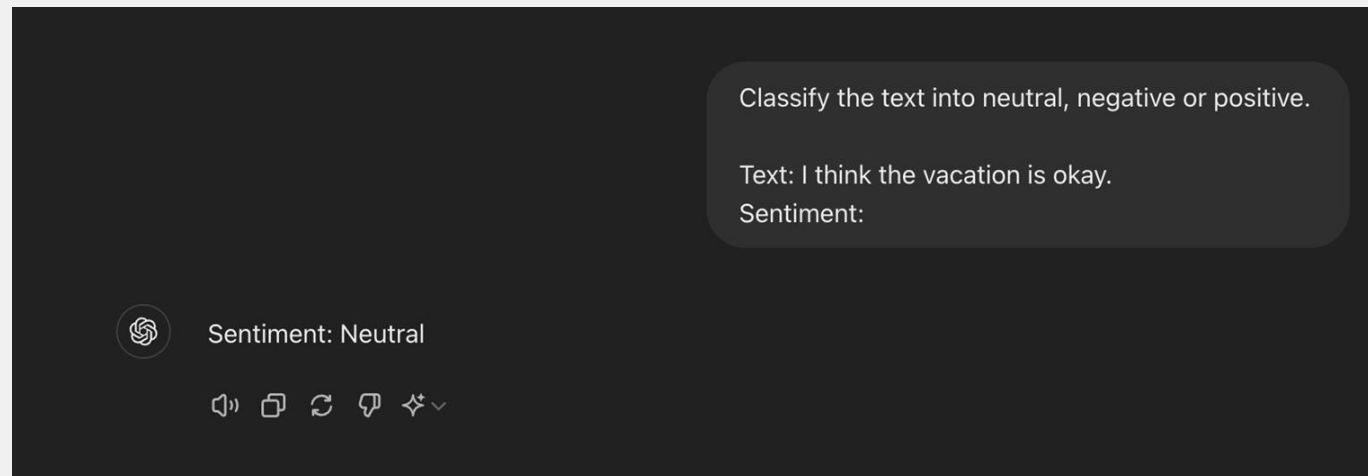
- **Zero-shot Prompting** → Utilizing generic prompts without prior examples.
- **Few-shot Prompting** → Providing a few examples to set a response pattern.
- **Fine-tuning with Examples** → Intensive example-based training to refine model understanding.

**Impact of Prompt Engineering** → Enhanced model performance, reliability, and domain-specific accuracy.



# ► Examples : Zero Shot Prompting

- Large-scale training makes these models capable of performing some tasks in a “**zero-shot**” manner.
- Zero-shot prompting means that the prompt used to interact with the model **won't contain examples** or demonstrations.
- The zero-shot prompt directly instructs the model to perform a task **without any additional examples to steer it**.



# ▶ Examples: Few Shot Prompting

- Few-shot prompting can be used as a technique to **enable in-context learning** where we provide demonstrations in the prompt to steer the model to better performance.
- The demonstrations serve as **conditioning for subsequent examples** where we would like the model to generate a response.

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:



Sure, here's an example sentence using the word "farduddle":

The children were so excited about their new puppy that they couldn't stop farduddling in the living room.



# Chain of Thoughts

- chain-of-thought (CoT) prompting enables **complex reasoning capabilities through intermediate reasoning steps**.
- You can **combine it with few-shot prompting** to get better results on more complex tasks that require reasoning before responding.

[Source](#)







# Chain of Thoughts

## 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. ✅

[Source](#)



# Chain of Thoughts

The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1. True or False ?



True.

The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.

A: Adding all the odd numbers (9, 15, 1) gives 25. The answer is False.

The odd numbers in this group add up to an even number: 17, 10, 19, 4, 8, 12, 24.

A: Adding all the odd numbers (17, 19) gives 36. The answer is True.

The odd numbers in this group add up to an even number: 16, 11, 14, 4, 8, 13, 24.

A: Adding all the odd numbers (11, 13) gives 24. The answer is True.

The odd numbers in this group add up to an even number: 17, 9, 10, 12, 13, 4, 2.

A: Adding all the odd numbers (17, 9, 13) gives 39. The answer is False.

The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.

A:



Adding all the odd numbers (15, 5, 13, 7, 1) gives 41. The answer is False.

[Source](#)





# ReACT

## (Reason + Act)

### (1d) ReAct (Reason + Act)

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.

**Act 1:** `Search[Apple Remote]`

**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the `Front Row media center` program ...

**Thought 2:** Apple Remote was originally designed to control the `Front Row media center` program. I need to search `Front Row` next and find what other device can control it.

**Act 2:** `Search[Front Row]`

**Obs 2:** Could not find `[Front Row]`. Similar: `['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]`

**Thought 3:** `Front Row` is not found. I need to search `Front Row (software)` .

**Act 3:** `Search[Front Row (software)]`

**Obs 3:** `Front Row` is a discontinued media center software ...

**Thought 4:** `Front Row (software)` is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

**Act 4:** `Finish[keyboard function keys]`



# ReACT Prompting

Chain-of-thought (CoT) prompting has shown the capabilities of LLMs to carry out reasoning traces to generate answers to questions involving arithmetic and commonsense reasoning, among other tasks. But its lack of access to the external world or inability to update its knowledge can lead to issues like fact hallucination and error propagation.

- ReAct is a general paradigm that **combines reasoning and acting** with LLMs. ReAct prompts LLMs to generate verbal reasoning traces and actions for a task.
- This allows the system to **perform dynamic reasoning to create, maintain, and adjust plans for acting** while also enabling interaction to external environments (e.g., Wikipedia) to incorporate additional information into the reasoning.



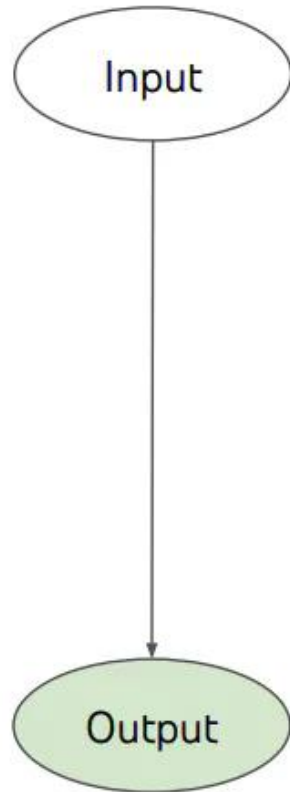
# Tree of Thoughts

- For complex tasks that require exploration or strategic lookahead, traditional or simple prompting techniques fall short.
- Tree of Thoughts (ToT) **generalizes over chain-of-thought prompting** and encourages exploration over thoughts that serve as intermediate steps for general problem solving with language models.
- ToT maintains a tree of thoughts, where **thoughts represent coherent language sequences** that **serve as intermediate steps** toward solving a problem.
- This approach **enables an LM to self-evaluate the progress through intermediate thoughts** made towards solving a problem through a deliberate reasoning process.

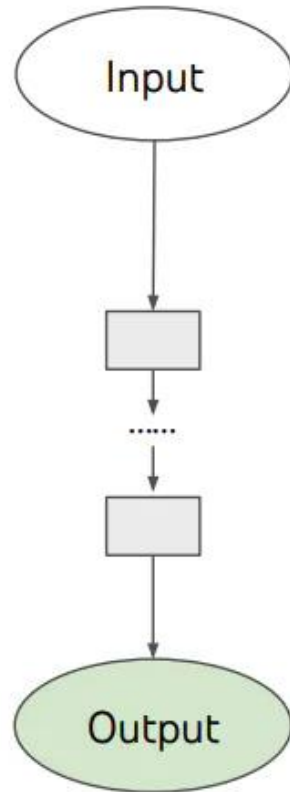




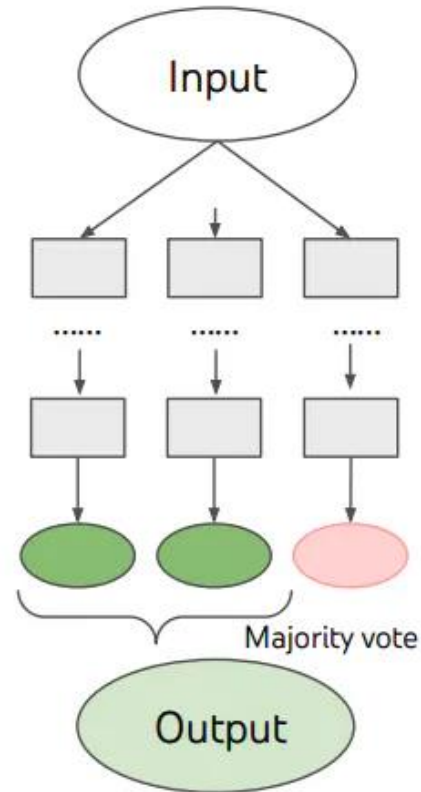
# Tree of Thoughts



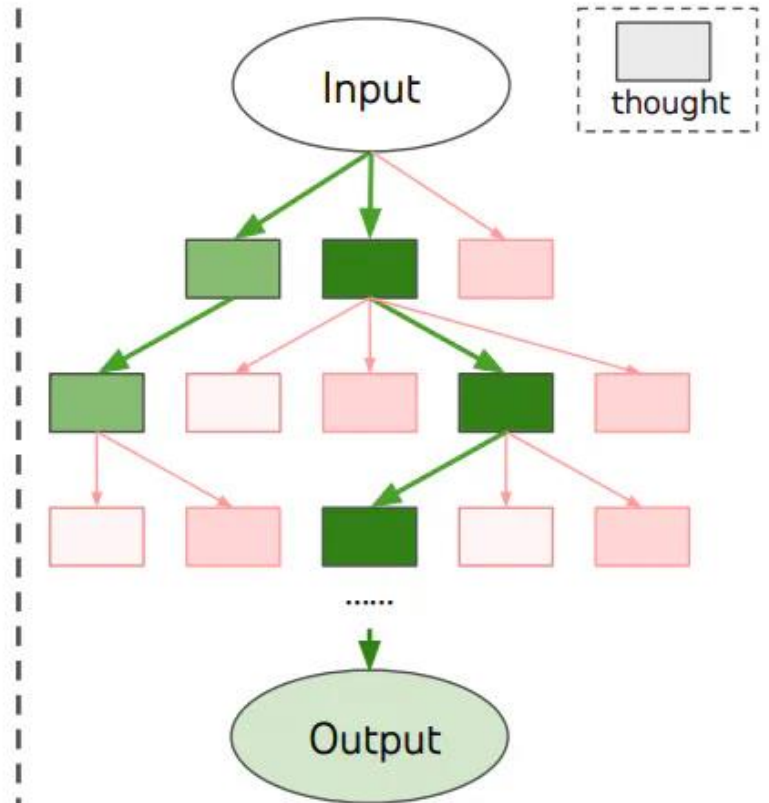
(a) Input-Output Prompting (IO)



(c) Chain of Thought Prompting (CoT)



(c) Self Consistency with CoT (CoT-SC)

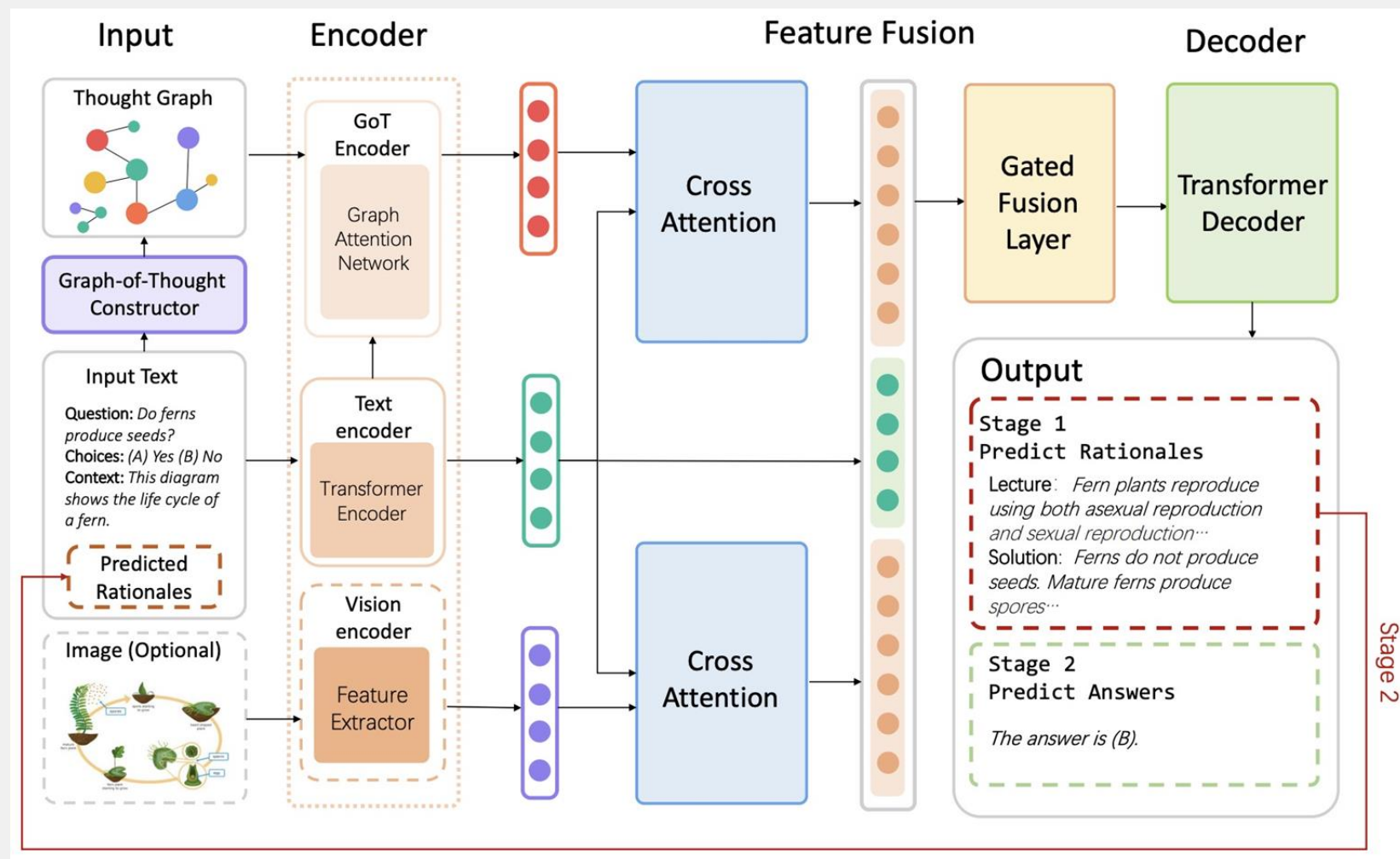


(d) Tree of Thoughts (ToT)





# Graph Of Thoughts



Stage 2



# Graph of Thoughts

- Human thought processes often follow non-linear patterns, deviating from simple sequential chains.
- Graph-of-Thought (GoT) reasoning makes model think as graphs rather than chains.
- **Paradigm Shift:**
  - Nodes in the graph represent thought units.
  - Edges depict connections, offering a realistic portrayal of complex human cognition.
  - GoT uses Directed Acyclic Graphs (DAGs), allowing paths that fork and converge, unlike traditional trees.
  - This approach provides significant advantages over conventional linear methods.
- **GoT Reasoning Model:**
  - Operates in a two-stage framework: generating rationales first, then producing the final answer.
  - Utilizes a Graph-of-Thoughts encoder for representation learning.
  - Integrates GoT representations with the original input through a gated fusion mechanism.
  - Combines both linear and non-linear aspects of thought processes.

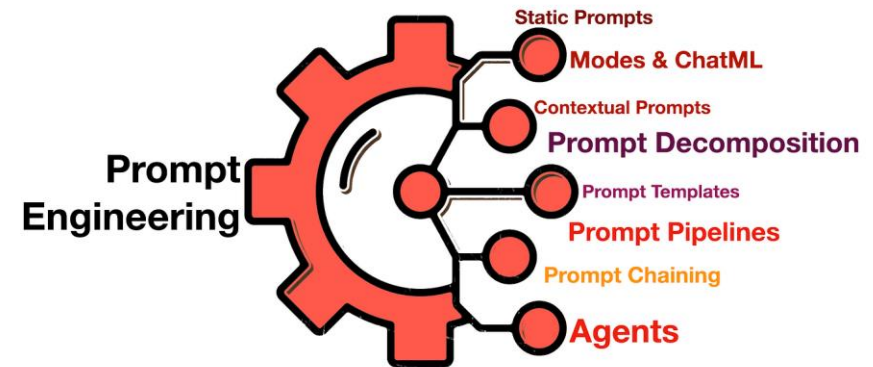




# ▶ Finishing with Prompt Engineering...

Prompt Engineering requires **deep** understanding of both the domain and the AI's capabilities.

Best practices are to use clear and concise languages, separate instruction and context clearly and to regularly update and test prompts to adapt to new domain knowledge and feedback.





# How to train your LLM?



# Introduction to LLM training

LLMs like GPT-4, GPT-3.5 and Claude are trained in a structured process to understand and generate human-like language. This training involves multiple phases, each designed to enhance the model's language capabilities and its understanding of human queries.



# Pre-Training Overview

In the pre-training phase, LLMs are trained to predict the next word in a sequence based on a massive dataset gathered from the internet. This foundational training uses diverse text sources to expose the model to a wide range of language patterns.



# Pre-Training: Details

During pre-training, the dataset is first cleaned and preprocessed to remove noise and irrelevant information. The text is then tokenized into smaller units using methods like Byte-Pair Encoding or WordPiece, essential for handling sequential data efficiently in transformer architectures.



# Pre-Training: Objective and Output

The primary goal of pre-training is to enable the LLM to predict the next word in a text sequence, helping it to start mimicking human language patterns. However, at this stage, the model only learns from the data it sees and doesn't understand context or specific user instructions.



# Supervised Fine-Tuning

Post pre-training, the model undergoes Supervised Fine-Tuning or Instruction Tuning where it is trained with specific user messages as inputs and AI-generated responses as targets. This phase helps the model to understand and generate appropriate responses based on the instructions provided.



# Output and Learning

After instruction tuning, the LLM can relate queries to correct responses, as seen in training examples like identifying capitals from questions. This step significantly enhances the model's ability to handle direct queries and provide accurate answers.







**What if this process  
doesn't generate a good  
enough model?**





# Enter RLHF: Reinforcement Learning from Human Feedback



# Overview of RLHF

Despite improvements, LLMs can still generate inappropriate or irrelevant responses. RLHF is introduced to refine the model's responses, ensuring they are helpful, honest, and harmless, aligning more closely with human values and practical utility.



# Implementing RLHF

RLHF starts with using a trained reward model, which evaluates multiple outputs from the LLM and ranks them based on human feedback. This model captures human preferences and is used to further fine-tune the LLM's responses.



# Large Scale training with Rewards Model

Once the reward model is sufficiently trained, it replaces human feedback in the loop, allowing for scaling up the fine-tuning process. This model guides the LLM in generating content that aligns better with human ethical standards and practical expectations.



# Tutorial

LangChain\_Tutorial.ipynb  
LangChain\_Exercise.ipynb



# Thank You!



**SDAIA**  
الهيئة السعودية للبيانات  
والذكاء الاصطناعي  
Saudi Data & AI Authority