



# IA para DEVS

---

Orquestrando Modelos de Linguagem

# Marcos Natã

## BACKGROUND

---



01

Bacharel em  
Engenharia de  
Computação

02

MBA em Gestão  
Empresarial

03

Pós em Machine  
Learning  
Engineering

04

2+ anos trabalhando  
com clientes  
americanos

05

Liderança na  
construção de  
produtos de IA do zero  
ao Go-To-Market

06

Gestão de times  
de engenheiros  
nos Estados  
Unidos e Índia

07

Grande experiência  
na indústria brasileira  
trabalhando com ERP

08

Empreendedor  
apaixonado por  
tecnologia

# MARCOS NATÃ

[lambdaly.ai](#)



[lambdaly.ai](#)



[@marcosnataqs](#)



[@marcosnataqs](#)



[@marcosnataqs](#)



[marcos.nata@lambdaly.ai](mailto:marcos.nata@lambdaly.ai)



+55 62 99309-5562



# State of A I

---

01

[GitHub Octoverse](#)  
[2023 - Rise of AI](#)

02

[Aligning AI](#)  
[systems with](#)  
[human intent](#)

03

[DALL-E 2 - Text-](#)  
[To-Image model](#)

04

[Sora - Text-To-](#)  
[Video model](#)

05

[Microsoft Copilot](#)

06

[Copilot - AI +](#)  
[Mixed Reality](#)

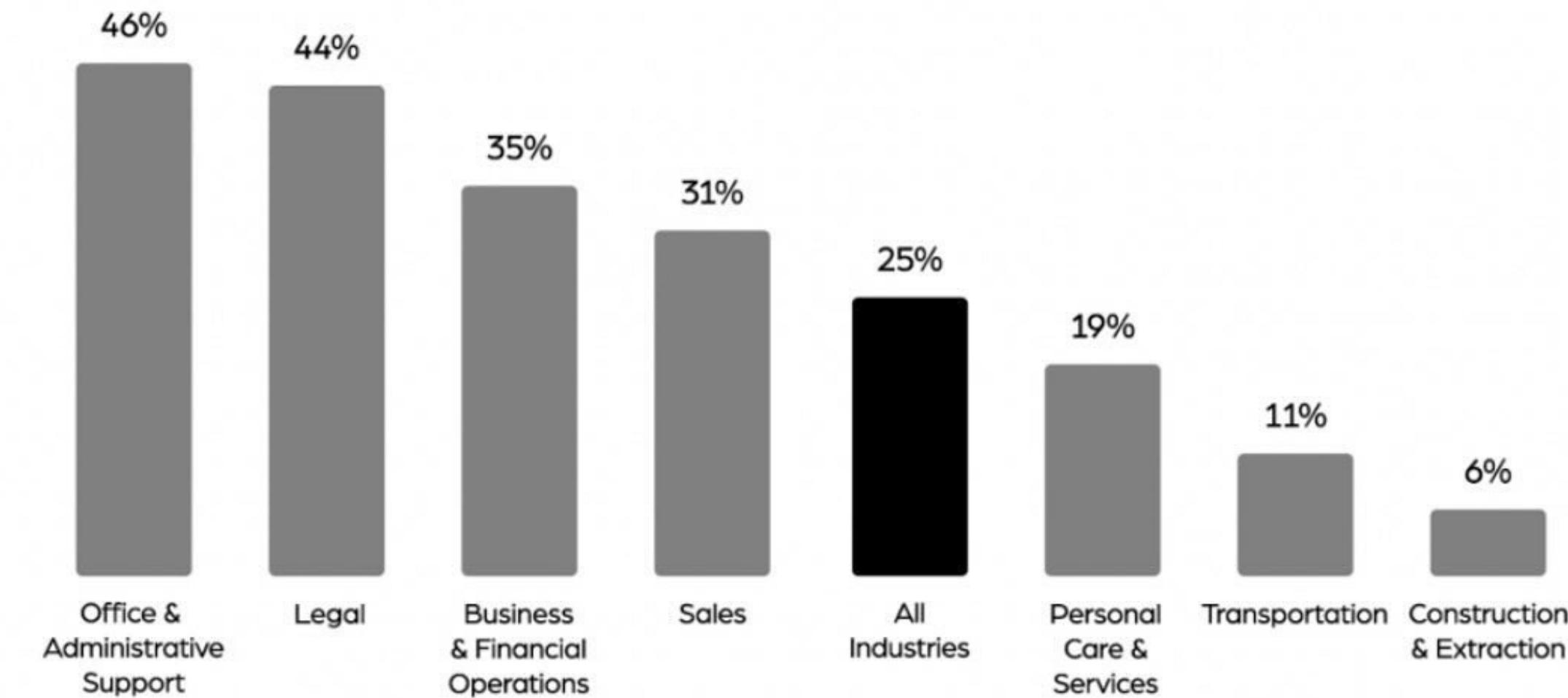
07

[AI Gadgets -](#)  
[Ballie](#)

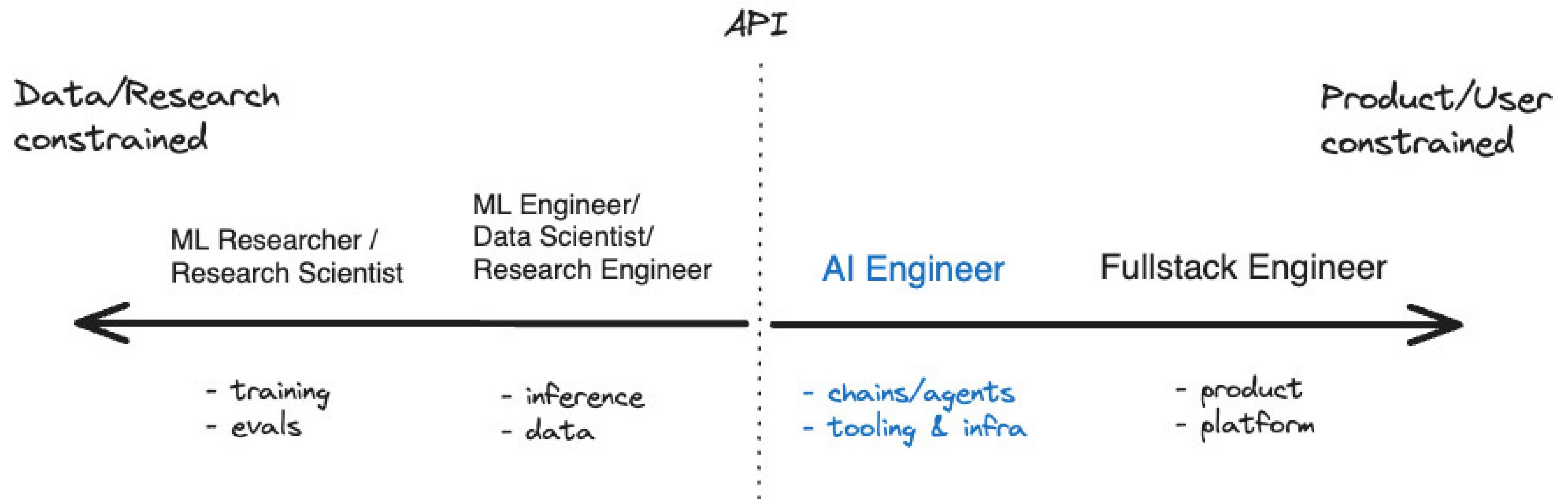
08

[Figure AI raises](#)  
[funding](#)  
[Robots - Figure AI](#)

## SHARE OF WORK THAT CAN BE AUTOMATED BY AI BY INDUSTRY



SOURCE: GOLDMAN SACHS



# IA para Devs PATHWAY

Bem-vindos ao curso "IA para Devs", a jornada definitiva para desenvolvedores que buscam mergulhar no revolucionário mundo da Inteligência Artificial aplicada à linguagem. Este curso é projetado para fornecer uma compreensão sólida e prática dos modelos de linguagem mais avançados, técnicas de engenharia de prompts e o uso efetivo de LangChain para a orquestração de modelos.

## Modelos de Linguagem

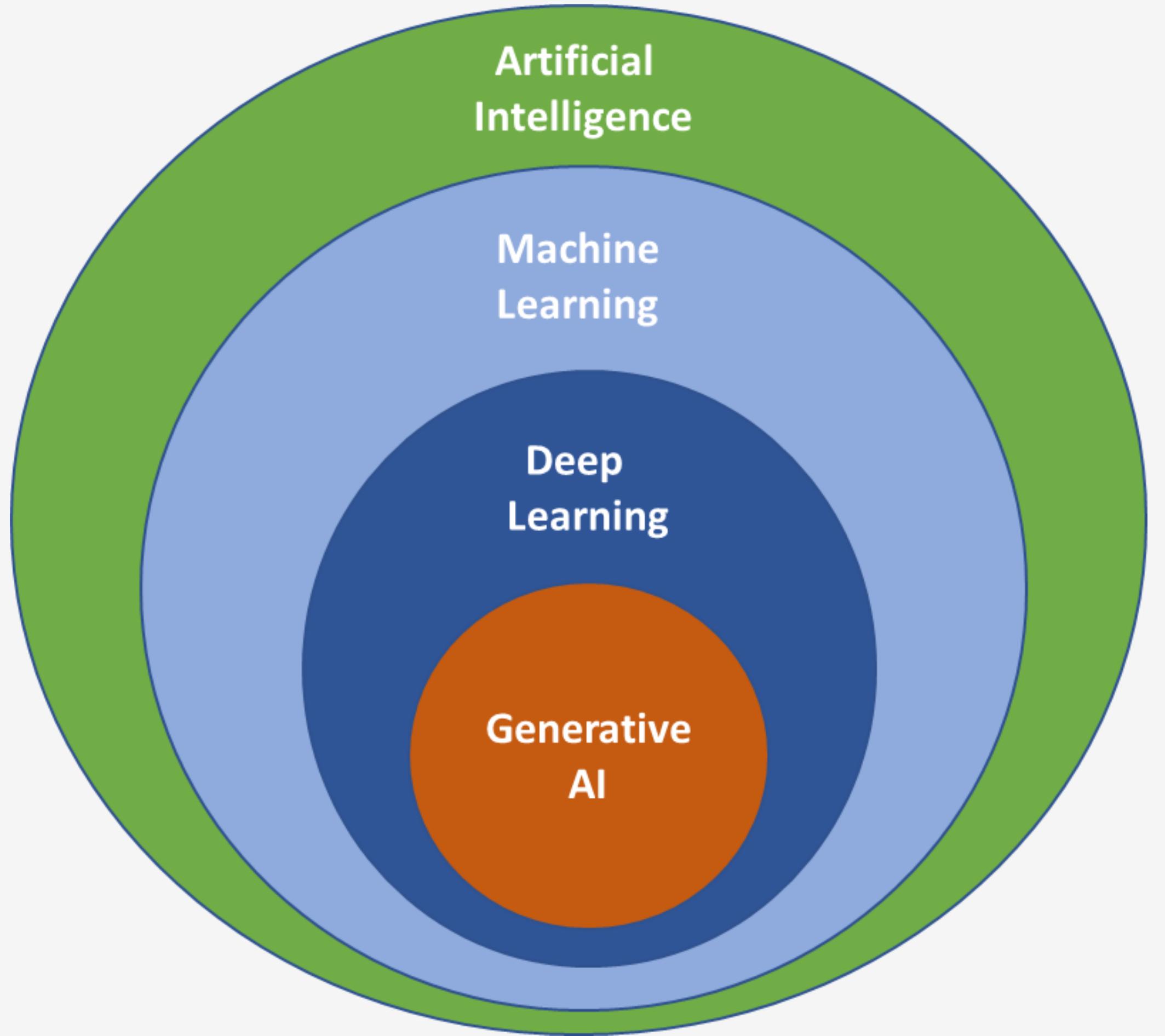
Descubra o que são modelos de linguagem, como eles são treinados e como podem ser aplicados para resolver problemas complexos do mundo real. Explore arquiteturas de IA de ponta e aprenda como personalizar e implementar esses modelos em seus próprios projetos de software.

## Prompt Engineering

Aprofunde-se na engenharia de prompts, uma habilidade essencial para quem deseja maximizar o desempenho dos modelos de linguagem. Aprenda a elaborar prompts claros, eficazes e criativos que direcionem a IA para gerar resultados precisos e úteis.

## LangChain

Explore o LangChain, um framework inovador para a orquestração de modelos de linguagem. Entenda como integrar diferentes modelos de IA para criar soluções complexas e personalizadas, superando as limitações de abordagens isoladas.



# Modelos de Linguagem

<https://artificialanalysis.ai/>

01

## OpenAI Models

<https://openai.com/>

02

## Anthropic Models

<https://www.anthropic.com/>

03

## Mistral Models

<https://mistral.ai/>

04

## Groq Models

<https://groq.com/>

# Responda as questões abaixo

**Oque é um System Prompt?**

**Oque é um User Prompt?**

**Oque é um temperatura de um modelo?**

**Oque é stop sequence?**

**Oque é o parâmetro Top P?**

**Oque é o parâmetro Frequency Penalty?**

**Oque é o parâmetro Presence Penalty?**

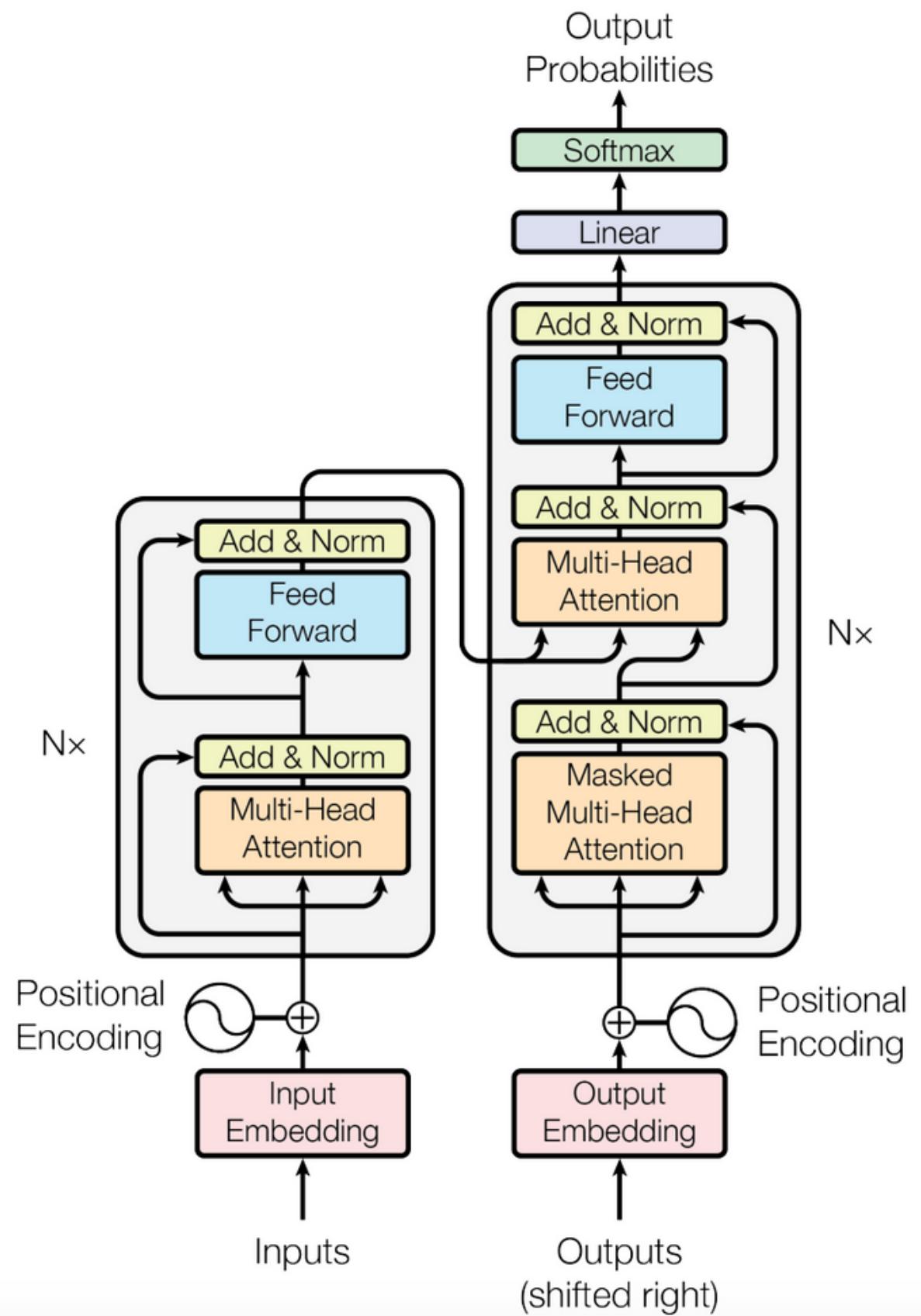
# **Hands-on: Vamos usar as APIs dos LLMs?**

# Como os modelos GPT funcionam?

1. **Coleta de Dados:** Compilação de textos variados da internet.
2. **Tokenização:** Divisão do texto em unidades menores (tokens).
3. **Arquitetura:** Baseada em Transformers, com ênfase no mecanismo de atenção.
4. **Treinamento:** Previsão de tokens subsequentes usando os anteriores.
5. **Ajuste Fino (Fine-Tunning):** Treinamento adicional em conjuntos de dados específicos.
6. **Avaliação:** Testes em várias tarefas para medir compreensão e desempenho.
7. **Objetivo:** Desenvolver um modelo capaz de entender e gerar texto natural de forma coerente.

# Arquitetura do Transformer

- **Base:** Revoluciona PLN com processamento paralelo e mecanismo de atenção.
- **Atenção Auto-dirigida:** Permite que o modelo considere todo o contexto da sequência simultaneamente.
- **Multi-Head Attention:** Captura relações complexas, observando diferentes partes da sequência.
- **Estrutura:** Empilhamento de blocos de codificadores e decodificadores.
  - **Codificadores:** Combinam atenção multi-cabeça e redes feed-forward.
  - **Decodificadores:** Adicionam atenção focada nas saídas do codificador.
- **Otimizações:** Conexões residuais e normalização para estabilidade no treinamento.
- **Vantagens:** Maior velocidade de treinamento e melhor qualidade em tarefas de PLN.



# Alguns Links

**The Illustrated GPT-2 (Visualizing Transformer Language Models)**

**How GPT3 Works - Visualizations and Animations**

**Attention Is All You Need**

**What are Generative AI models?**

# Prompt Engineering

01

**Zero-Shot Prompting**

02

**Few-Shot Prompting**

03

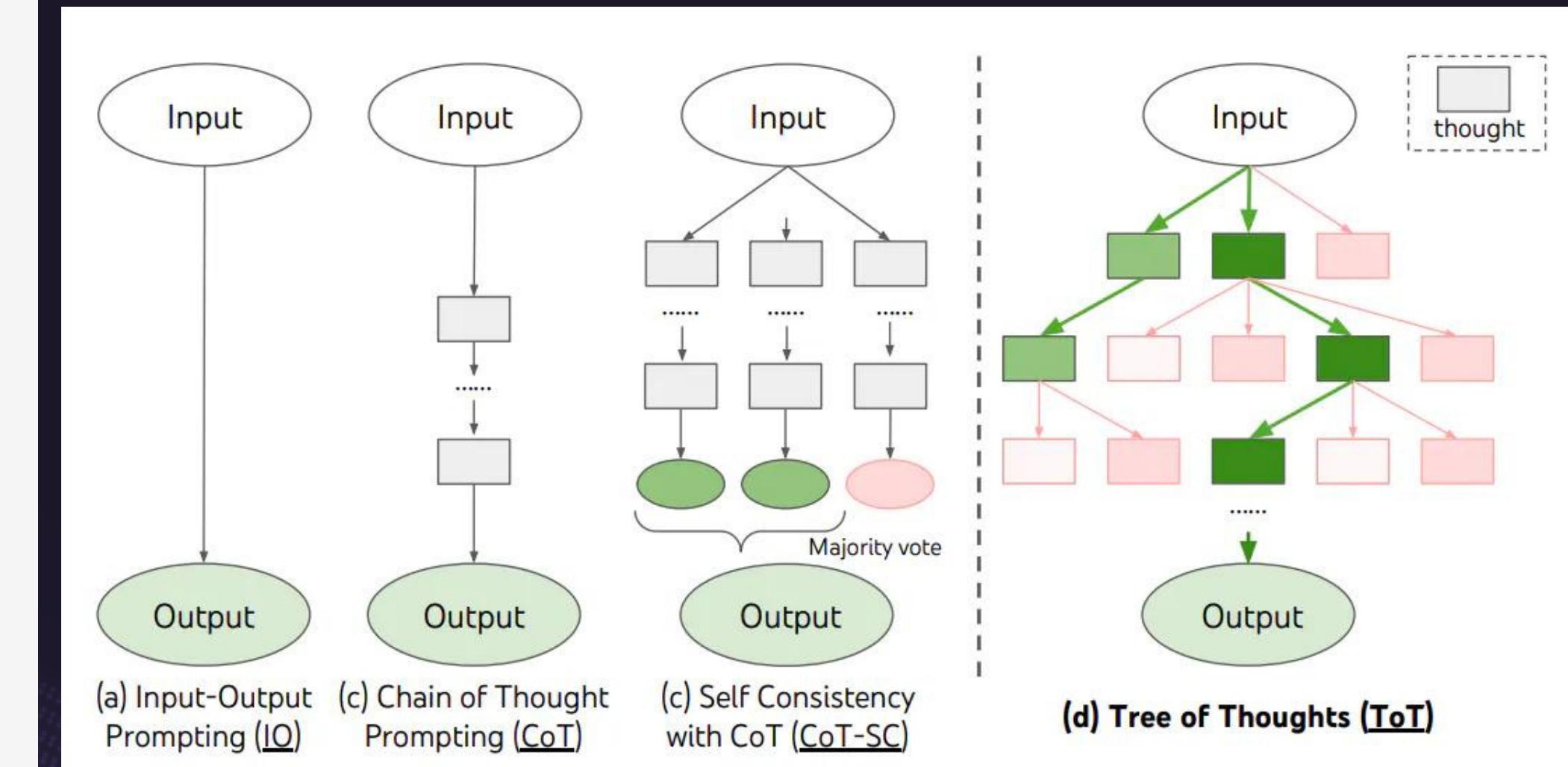
**Chain-of-Thought  
Prompting**

04

**Self-Consistency**

# Tree of Thoughts (ToT)

ToT mantém uma árvore de pensamentos, onde os pensamentos representam sequências de linguagem coerentes que servem como etapas intermediárias para a resolução de um problema. Esta abordagem permite que um LM autoavalie o progresso através de pensamentos intermediários feitos para resolver um problema através de um processo de raciocínio deliberado. A capacidade do LM de gerar e avaliar pensamentos é então combinada com algoritmos de busca (por exemplo, breadth-first search e depth-first search) para permitir a exploração sistemática de pensamentos com antecipação e retrocesso.



# ReAct Prompting

ReAct (reason, act) é um paradigma para permitir que modelos de linguagem resolvam tarefas complexas usando raciocínio em linguagem natural. O ReAct foi projetado para tarefas nas quais o LLM pode realizar determinadas ações. Por exemplo, como em um sistema MRKL, um LLM pode interagir com APIs externas para recuperar informações. Quando questionado, o LLM pode optar por realizar uma ação para recuperar informações e, em seguida, responder à pergunta com base nas informações recuperadas.

## (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.

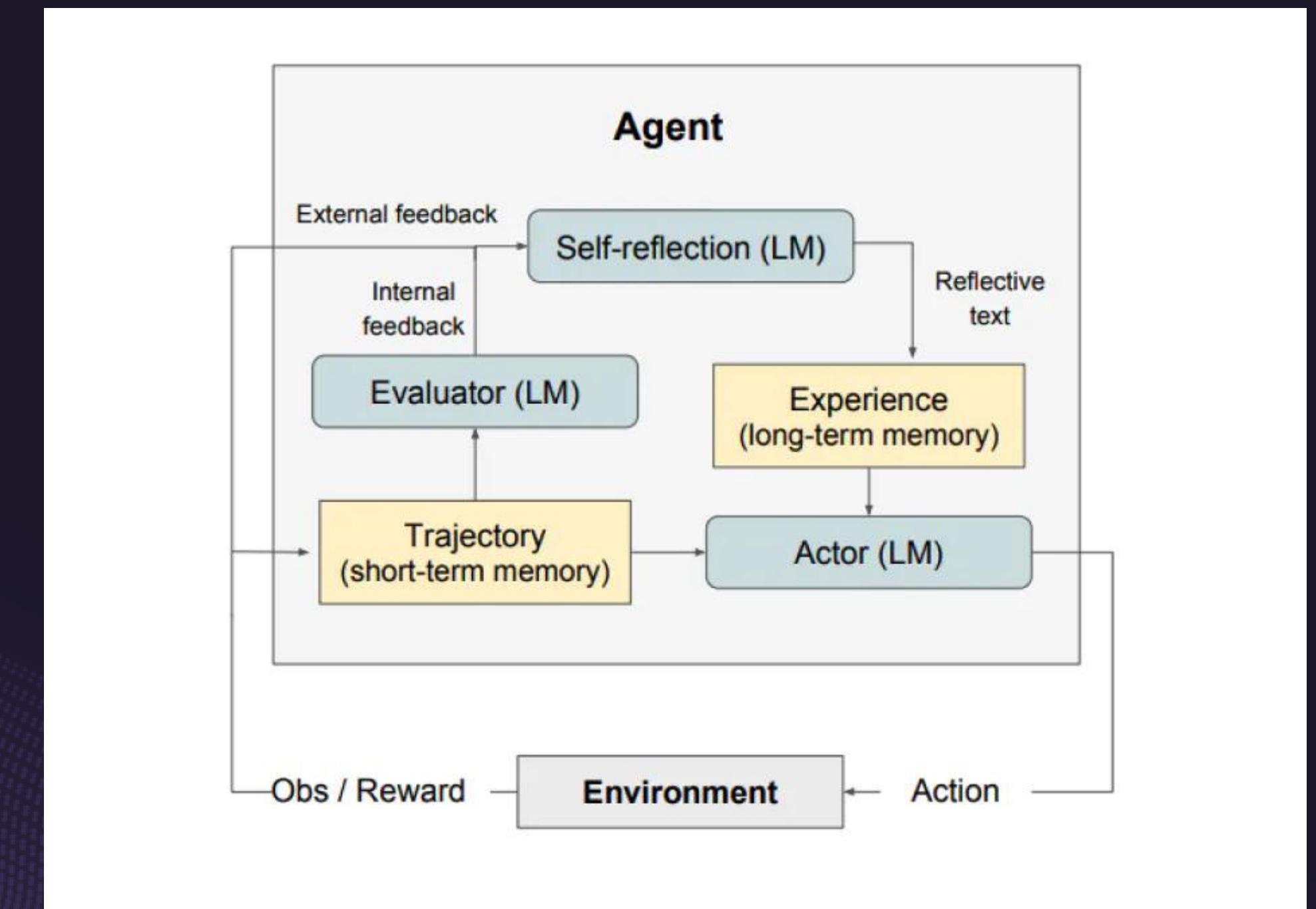
**Act 4:** `Finish[keyboard function keys]`



# Reflexion

A reflexão é uma estrutura para reforçar agentes baseados na linguagem através de feedback linguístico. De acordo com Shinn et al. (2023), "Reflexão é um novo paradigma para reforço 'verbal' que parametriza uma política como a codificação de memória de um agente emparelhada com uma escolha de parâmetros LLM."

Em um nível elevado, a Reflexão converte o feedback (linguagem de forma livre ou escalar) do ambiente em feedback linguístico, também conhecido como autorreflexão, que é fornecido como contexto para um agente LLM no próximo episódio. Isso ajuda o agente a aprender de forma rápida e eficaz com os erros anteriores, levando a melhorias de desempenho em muitas tarefas avançadas.



(a) **Task**  
 ↓  
 (b) **Trajectory**  
 ↓  
 (c) **Evaluation**  
 (internal / external)  
 ↓  
 (d) **Reflection**  
 ↓  
 (e) **Next Trajectory**

### 1. Decision making

You are in the middle of a room  
 [...] **Task:** clean some pan and put it in countertop.

### 2. Programming

**Task:** You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]

### 3. Reasoning

**Task:** What profession does John Lanchester and Alan Dean Foster have in common?

[...]  
**Action:** take pan1 from stoveburner1  
**Obs:** Nothing happens. [...]  
**Action:** clean pan1 with sinkbasin1  
**Obs:** Nothing happens. [...]

```
def match_parens(lst):
    if s1.count('(') + s2.count('(') == s1.count(')') + s2.count(')').count(')'): [...]
        return 'No'
```

**Think:** [...] novelist, journalist, critic [...] novelist, screenwriter [...] common is novelist and screenwriter.  
**Action:** "novelist, screenwriter"

**Rule/LM Heuristic:**  
 Hallucination.

**Self-generated unit tests fail:**  
`assert match_parens(...)`

**Environment Binary Reward:**  
 0

[...] tried to pick up the pan in stoveburner 1 [...] but the pan was not in stoveburner 1. [...]

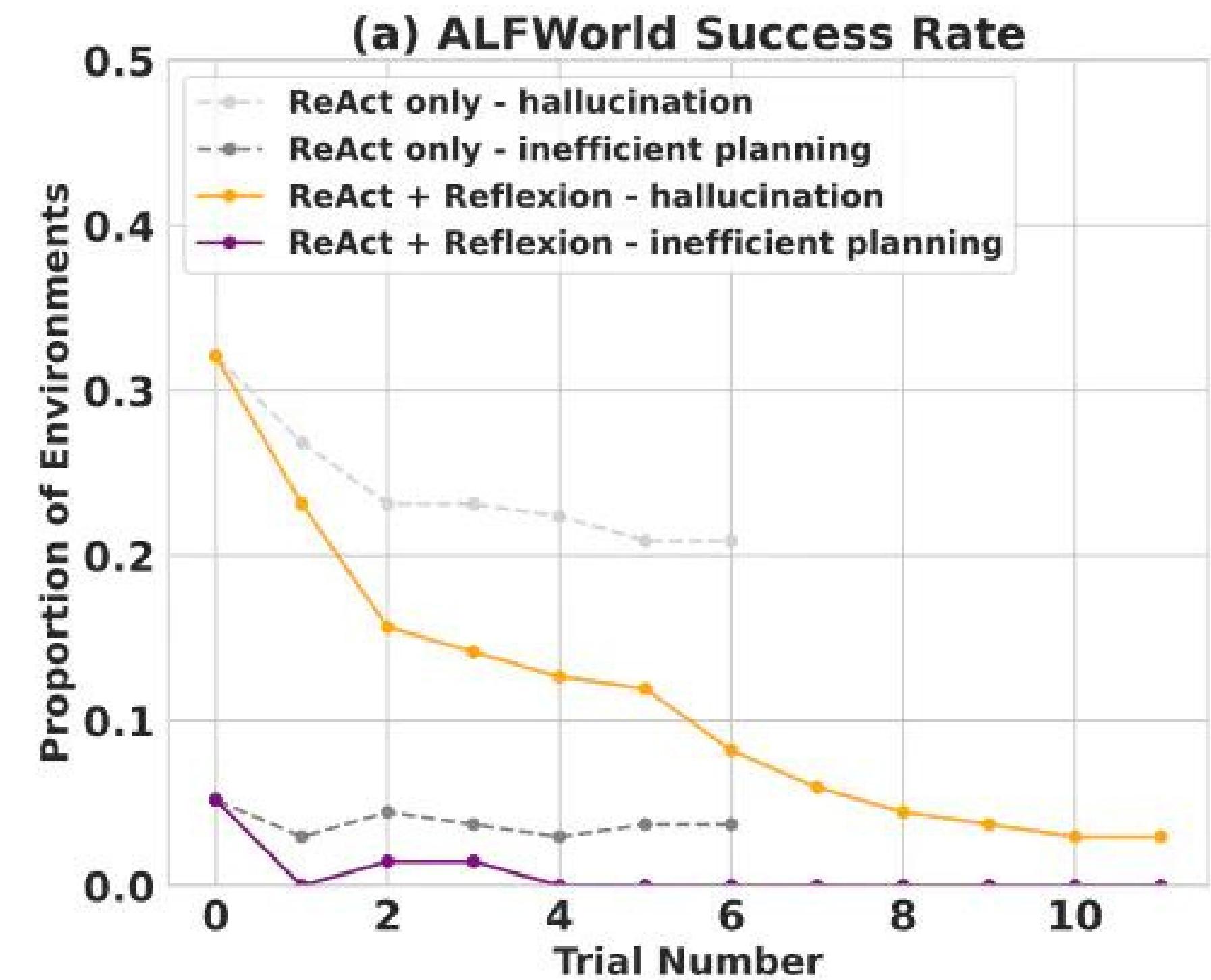
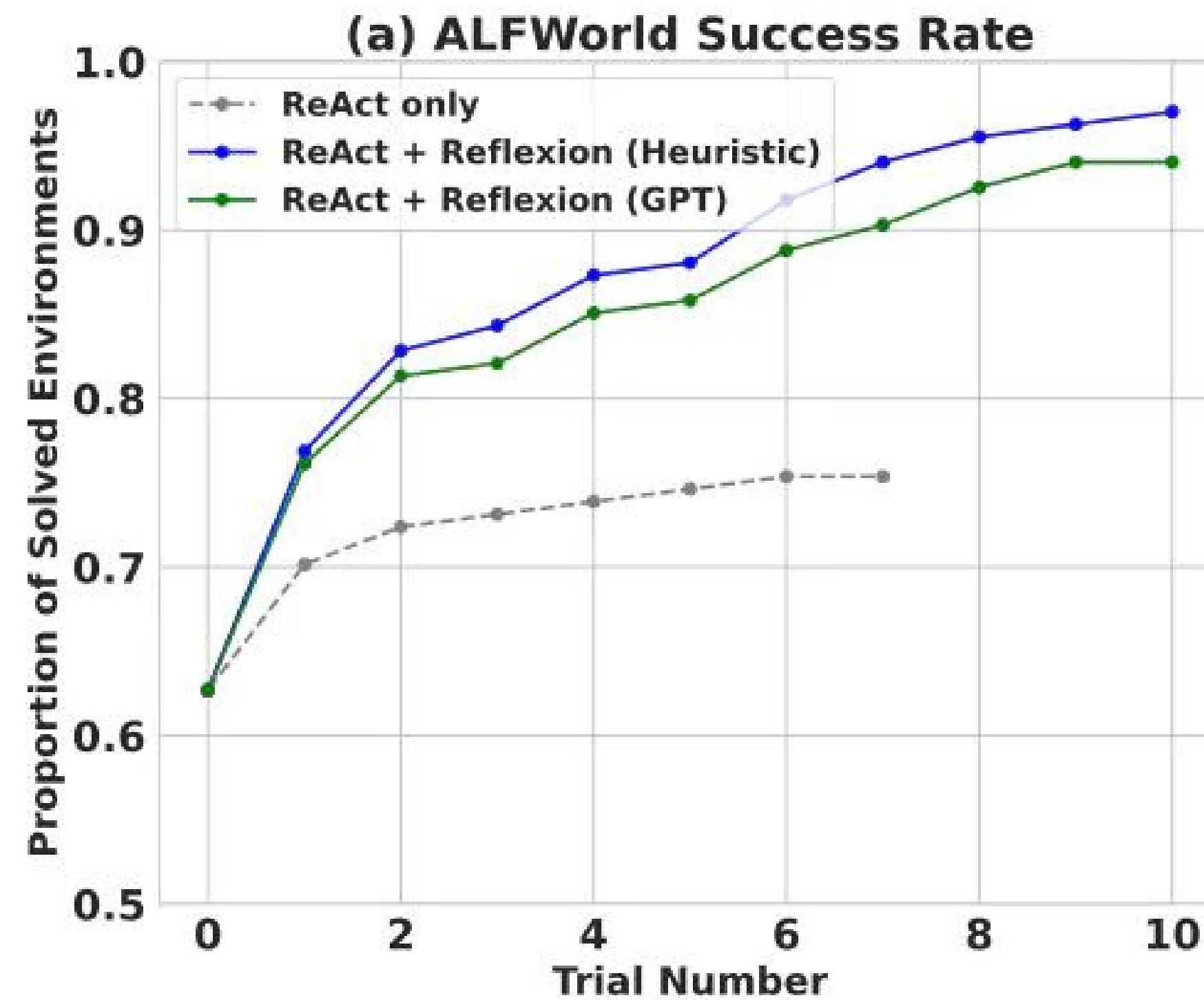
[...] wrong because it only checks if the total count of open and close parentheses is equal [...] order of the parentheses [...]

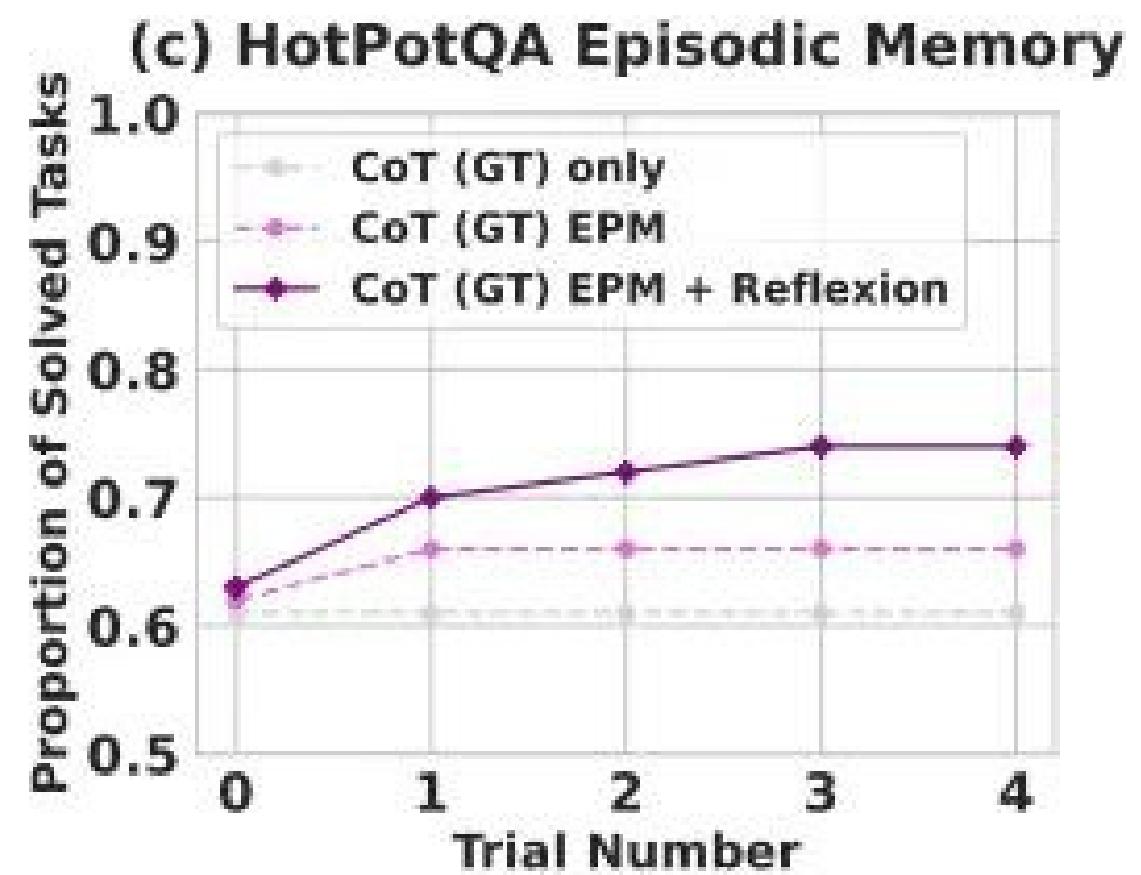
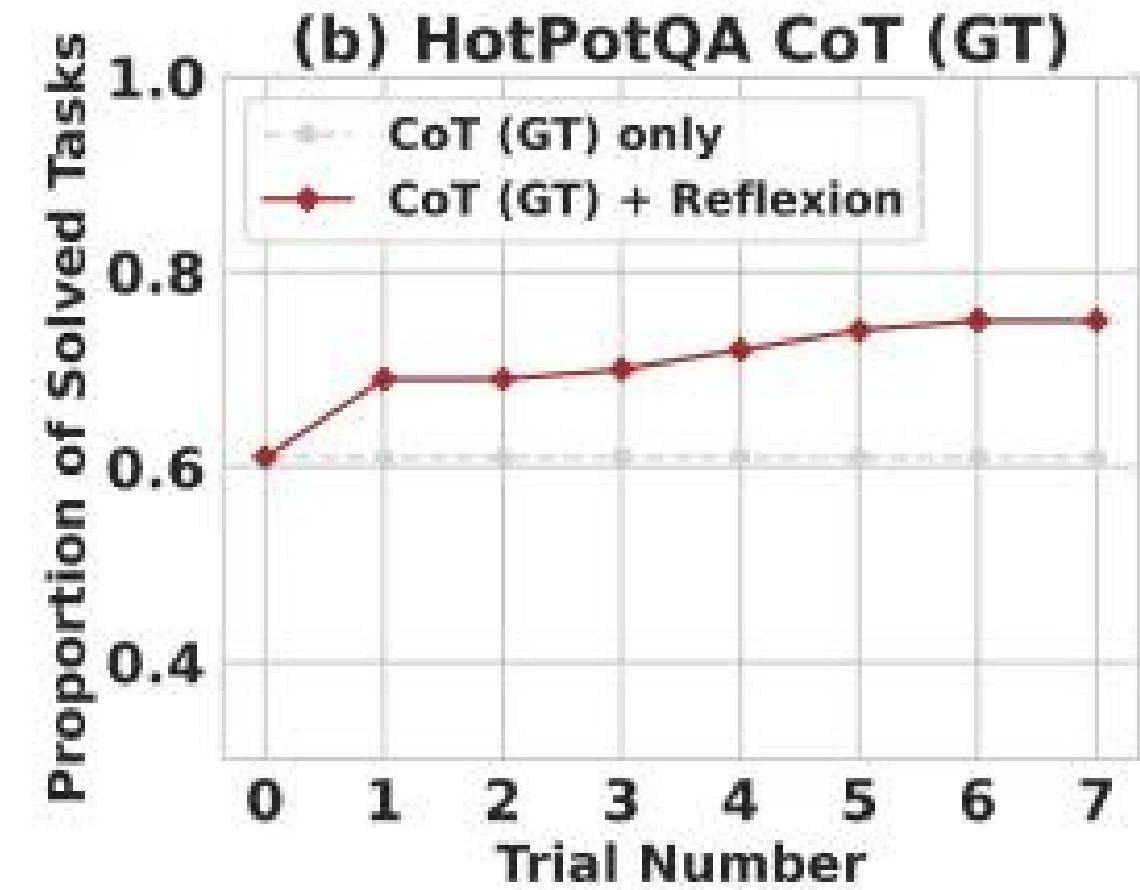
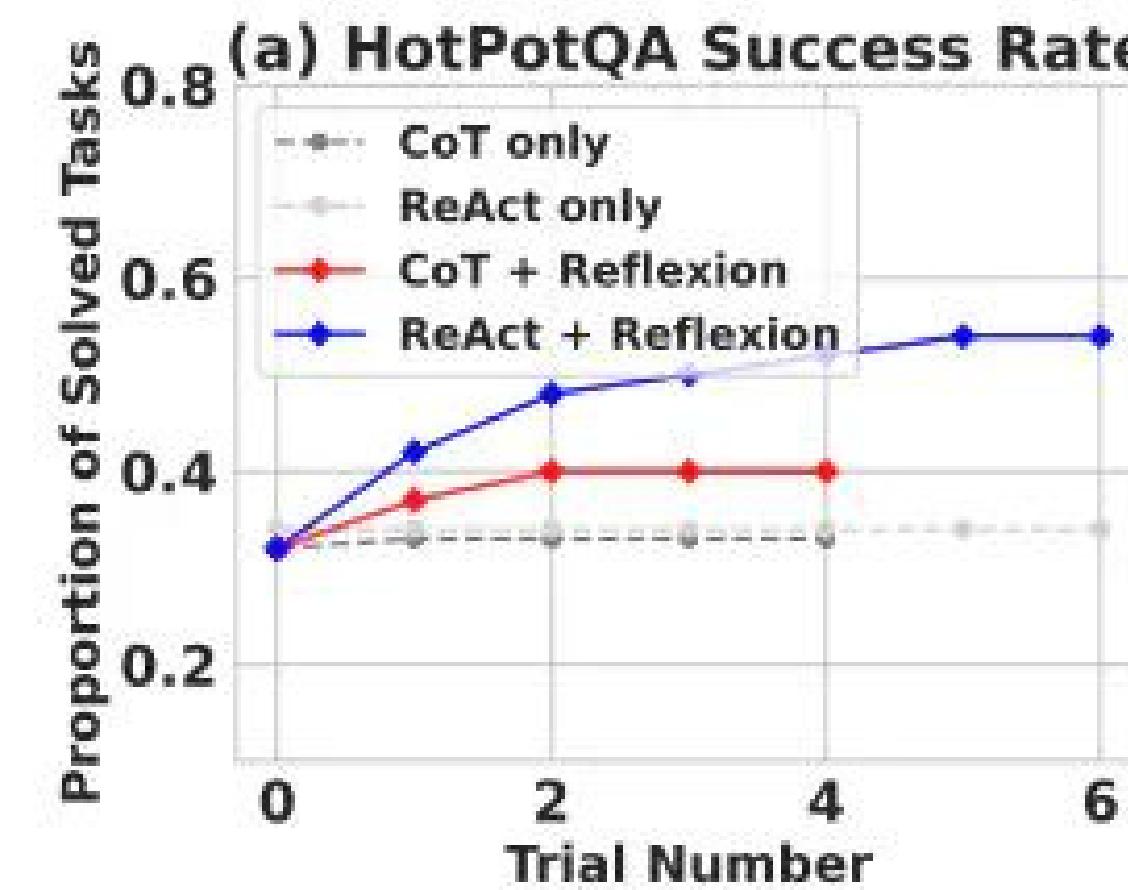
[...] failed because I incorrectly assumed that they both had the same multiple professions [...] accurately identifying their professions.

[...] **Action:** take pan 1 from stoveburner 2  
 [...] **Obs:** You put the pan 1 in countertop 1.

[...]  
`return 'Yes' if check(S1) or check(S2) else 'No'`

**Think:** [...] So the profession John Lanchester and Alan Dean Foster have in common is novelist.  
**Action:** "novelist"





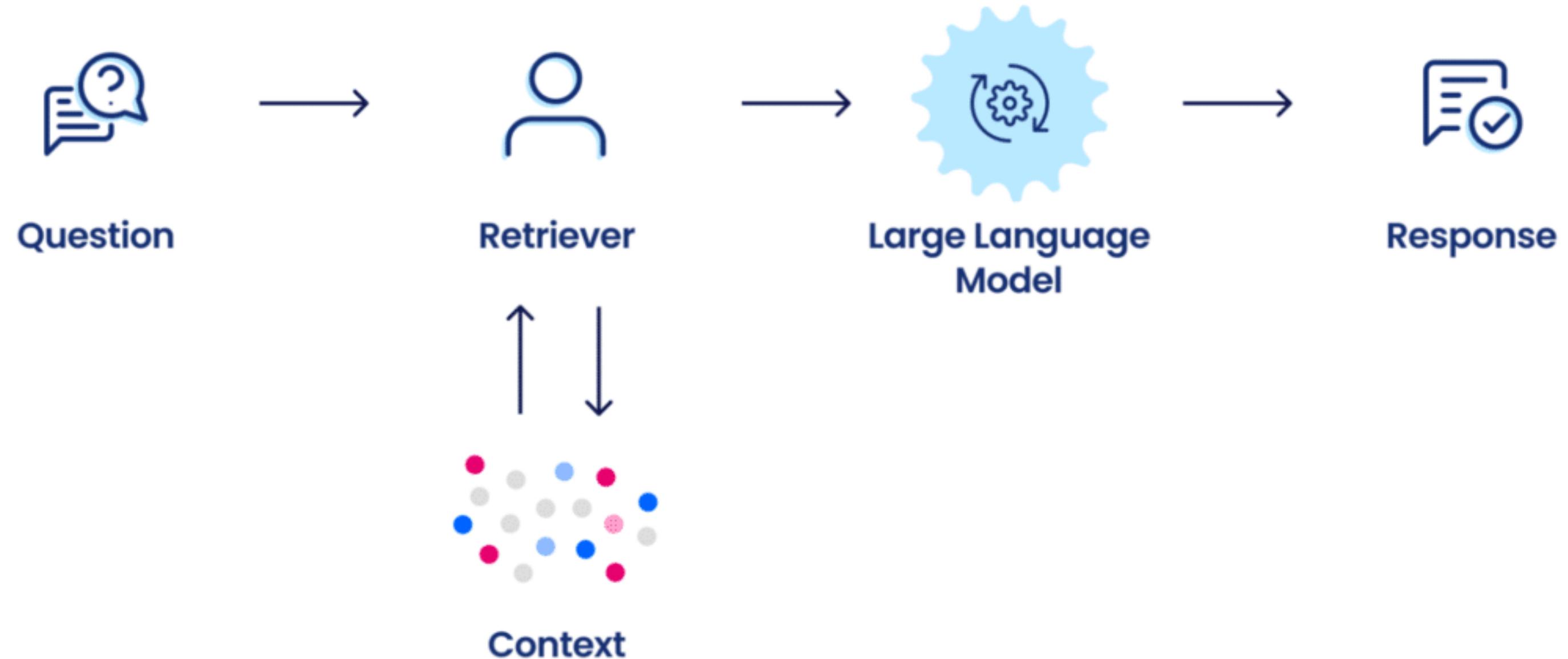
# Function Calling

Em uma chamada de API, você pode descrever funções e fazer com que o modelo escolha de forma inteligente a saída de um objeto JSON contendo argumentos para chamar uma ou mais funções. A API Chat Completions não chama a função; em vez disso, o modelo gera JSON que você pode usar para chamar a função em seu código.

Os modelos mais recentes (gpt-3.5-turbo-0125 e gpt-4-turbo-preview) foram treinados para detectar quando uma função deve ser chamada (dependendo da entrada) e para responder com JSON que segue a assinatura da função mais de perto do que os modelos anteriores.

```
{  
  "type": "function",  
  "function": {  
    "name": "get_current_weather",  
    "description": "Get the current weather in a given location",  
    "parameters": {  
      "type": "object",  
      "properties": {  
        "location": {  
          "type": "string",  
          "description": "The city and state, e.g. San Francisco, CA"  
        },  
        "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]}  
      },  
      "required": ["location"]  
    }  
  }  
}
```

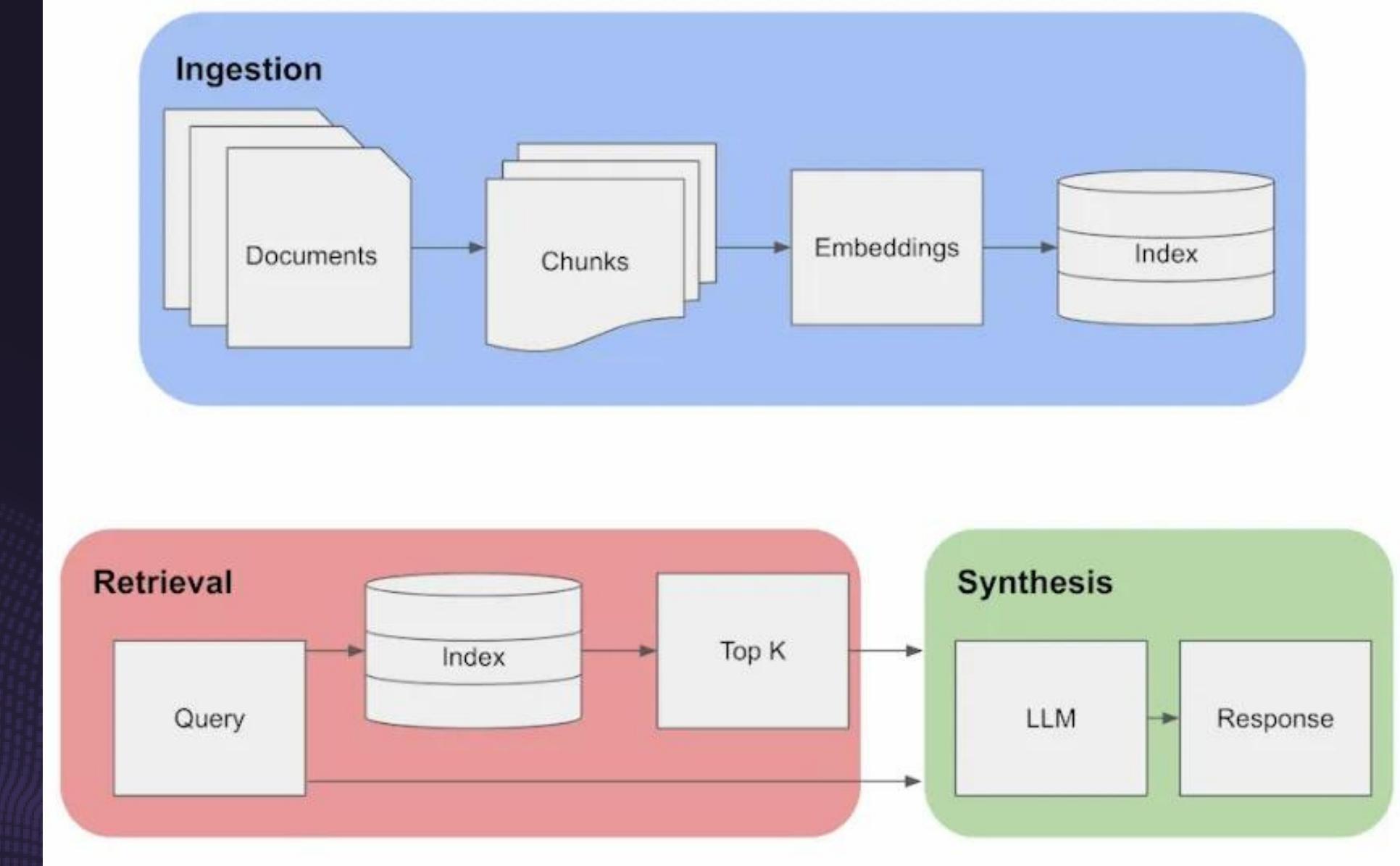
## Retrieval Augmented Generation



# Basic RAG Pipeline

Retrieval Augmented Generation (RAG) é uma técnica de processamento de linguagem natural (PNL) que melhora a saída de grandes modelos de linguagem (LLMs) usando dados personalizados. O RAG faz isso recuperando documentos relevantes e fornecendo-os como contexto para o LLM. O RAG utiliza fontes externas de conhecimento para concluir tarefas complexas e pode gerar respostas mais factuais, específicas e diversas.

## Basic RAG Pipeline



## Vector Stores

### 1. Load Source Data



Load, Transform, Embed

### Vector Store

0.5, 0.2....0.1, 0.9  
:  
2.1, 0.1....-1.7, 0.9

### 2. Query Vector Store

#### Embed

5.5, -0.3...  
2.1, 0.1

XXXXXXXXXXXXXX  
XXXXXXXXXXXXXX

XXXXXXXXXXXXXX  
XXXXXXXXXXXXXX

### 3. Retrieve 'most similar'

### Pure vector databases



Weaviate



Vald



### Vector-capable NoSQL databases



DataStax

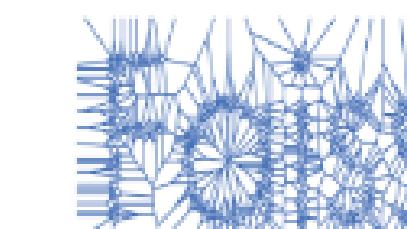
Astra



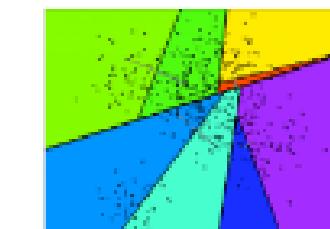
redis



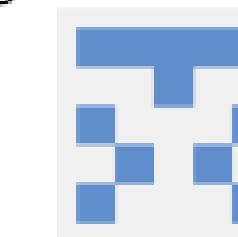
MongoDB



FAISS



Annoy



Hnswlib

### Text search databases



elasticsearch



### Vector-capable SQL databases

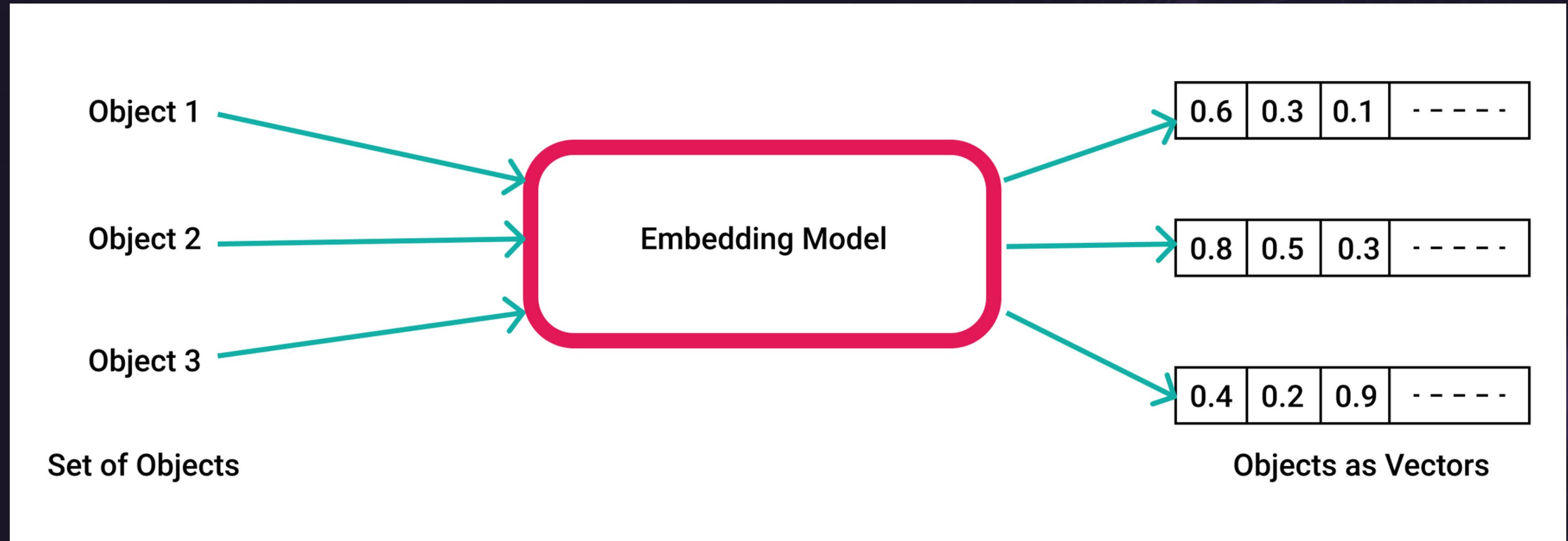


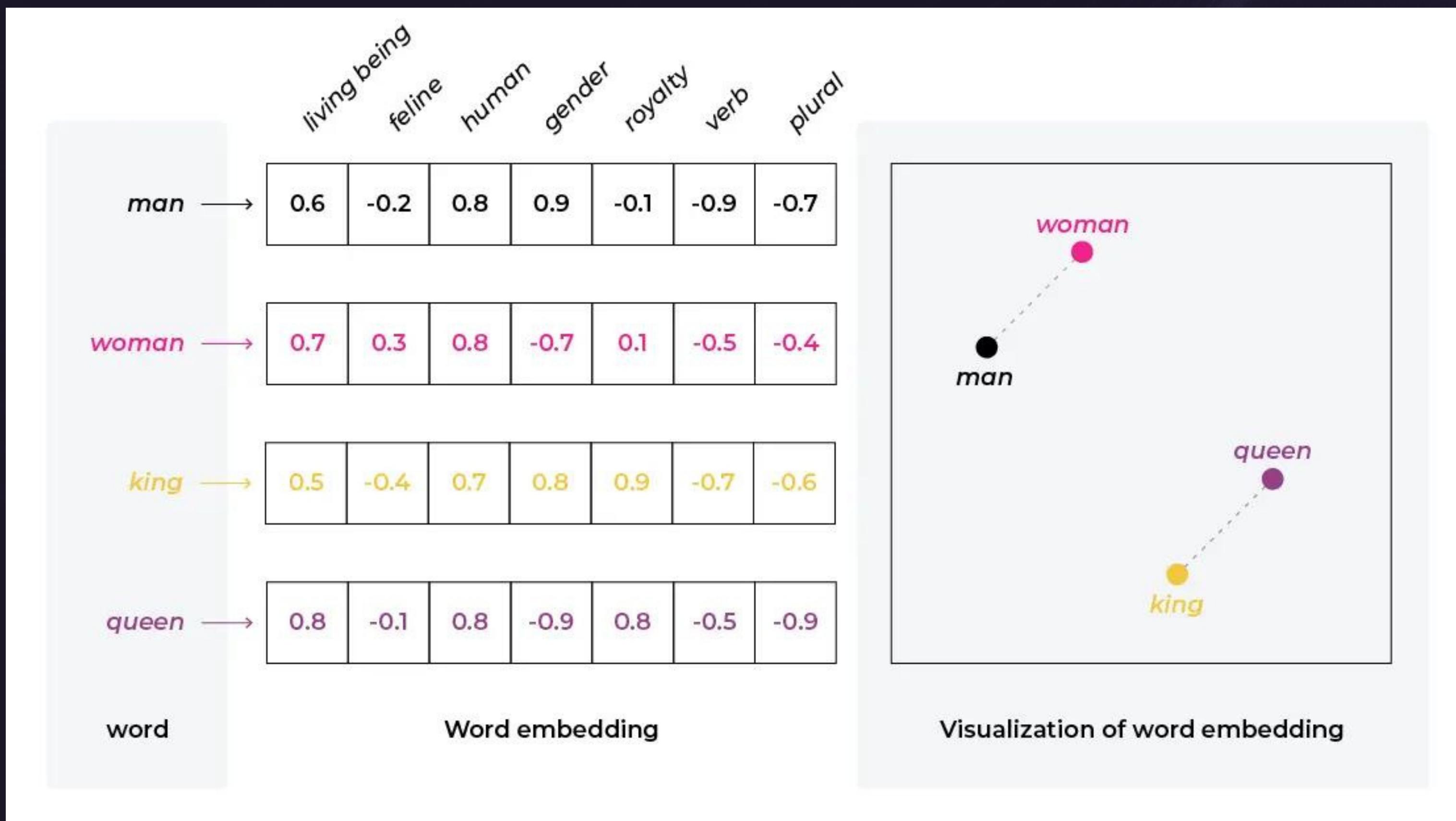
pgvector for Postgres



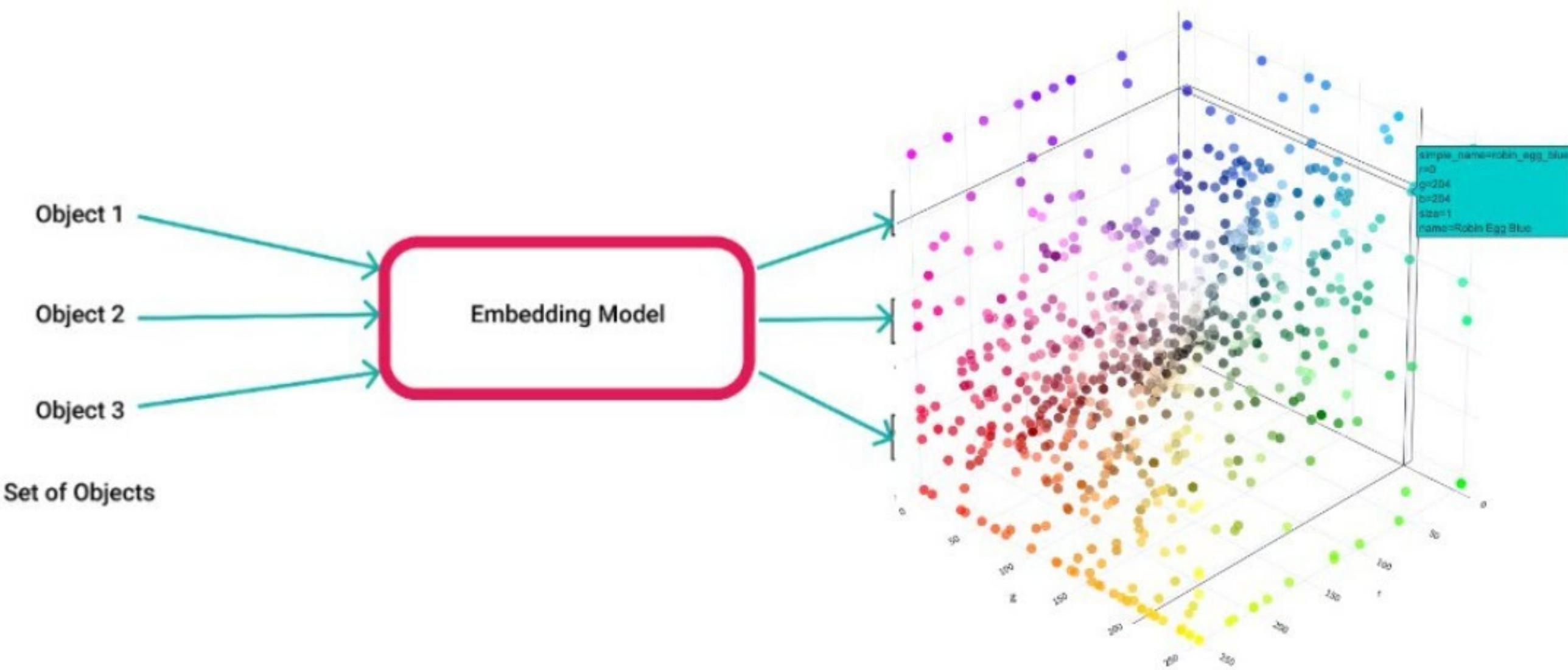
ClickHouse

kinetica [ROCKSET]





# INTRODUCTION TO VECTOR EMBEDDINGS



# Hands-on: Criar um Index em um banco vetorial (VectorDB)



Pinecone

# Armadilhas das LLMs

## Citando Fontes

Embora os LLMs possam gerar texto que parece citar fontes, é importante observar que eles não podem citar fontes com precisão. Isso ocorre porque eles não têm acesso à Internet e não conseguem lembrar de onde vieram seus dados de treinamento. Como resultado, muitas vezes geram fontes que parecem plausíveis, mas são inteiramente fabricadas. Esta é uma limitação significativa ao usar LLMs para tarefas que exigem citação precisa da fonte.

# Armadilhas das LLMs

## Viés

Os LLMs podem apresentar preconceitos em suas respostas, muitas vezes gerando conteúdo estereotipado ou preconceituoso. Isso ocorre porque eles são treinados em grandes conjuntos de dados que podem conter informações tendenciosas. Apesar das salvaguardas implementadas para evitar isto, os LLMs podem por vezes produzir conteúdo sexista, racista ou homofóbico. Esta é uma questão crítica a ter em conta ao utilizar LLMs em aplicações voltadas para o consumidor ou em investigação, pois pode levar à propagação de estereótipos prejudiciais e resultados tendenciosos.

# Armadilhas das LLMs

## Alucinações

Às vezes, os LLMs podem "alucinar" ou gerar informações falsas quando fazem uma pergunta para a qual não sabem a resposta. Em vez de afirmar que não sabem a resposta, muitas vezes geram uma resposta que parece confiante, mas incorreta. Isto pode levar à disseminação de desinformação e deve ser tido em conta ao utilizar LLMs para tarefas que requerem informações precisas.

# Armadilhas das LLMs

## Matemática

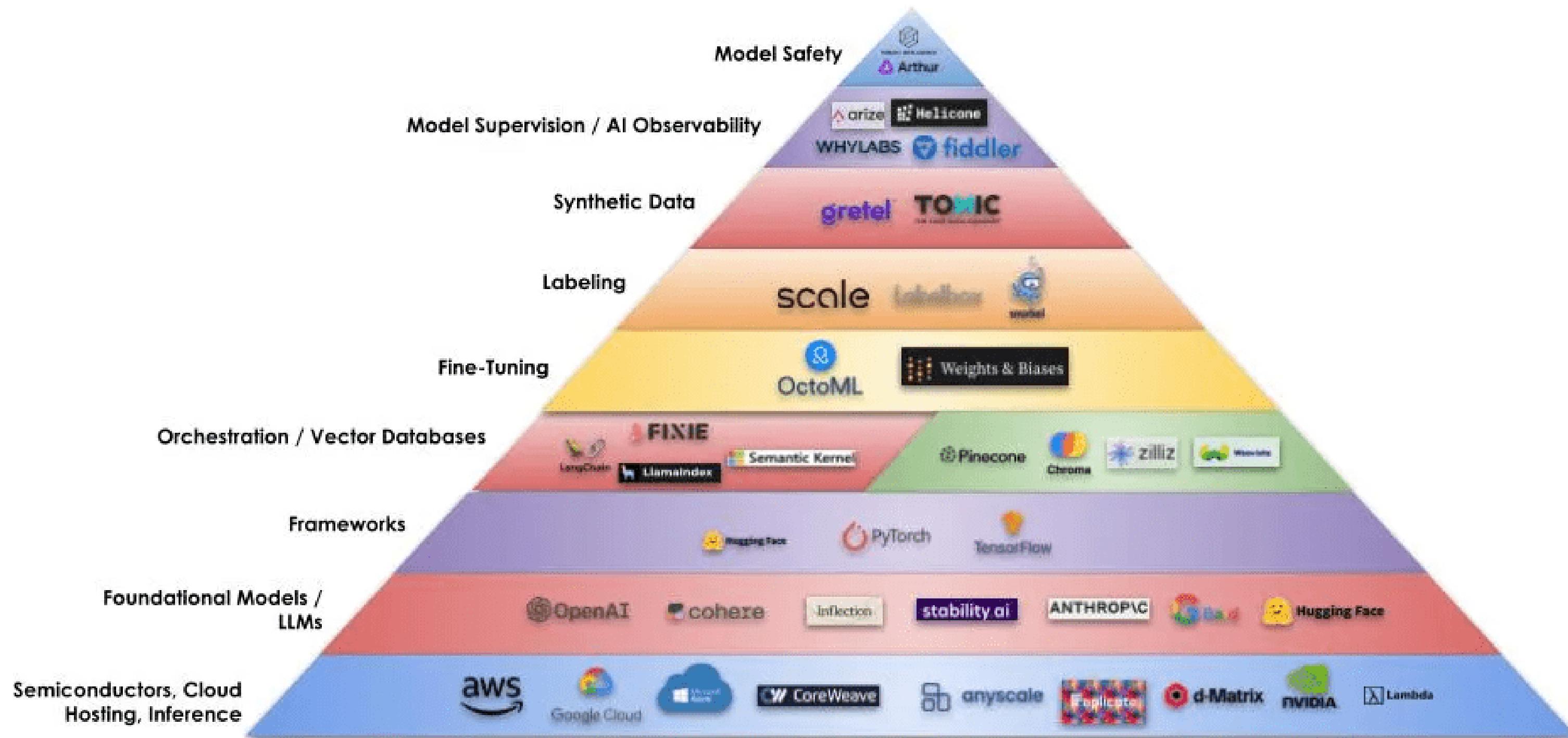
Apesar de suas capacidades avançadas, os Large Language Models (LLMs) muitas vezes enfrentam dificuldades com tarefas matemáticas e podem fornecer respostas incorretas (até mesmo tão simples quanto multiplicar dois números). Isso ocorre porque eles são treinados em grandes volumes de texto e a matemática pode exigir uma abordagem diferente.

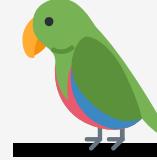
# Armadilhas das LLMs

## Prompt Hacking

Os LLMs podem ser manipulados ou “hackeados” pelos usuários para gerar conteúdo específico. Isso é conhecido como prompt hacking e pode ser usado para induzir o LLM a gerar conteúdo impróprio ou prejudicial. É importante estar ciente deste problema potencial ao usar LLMs, especialmente em aplicações voltadas para o público.

# The Building Blocks of Generative AI

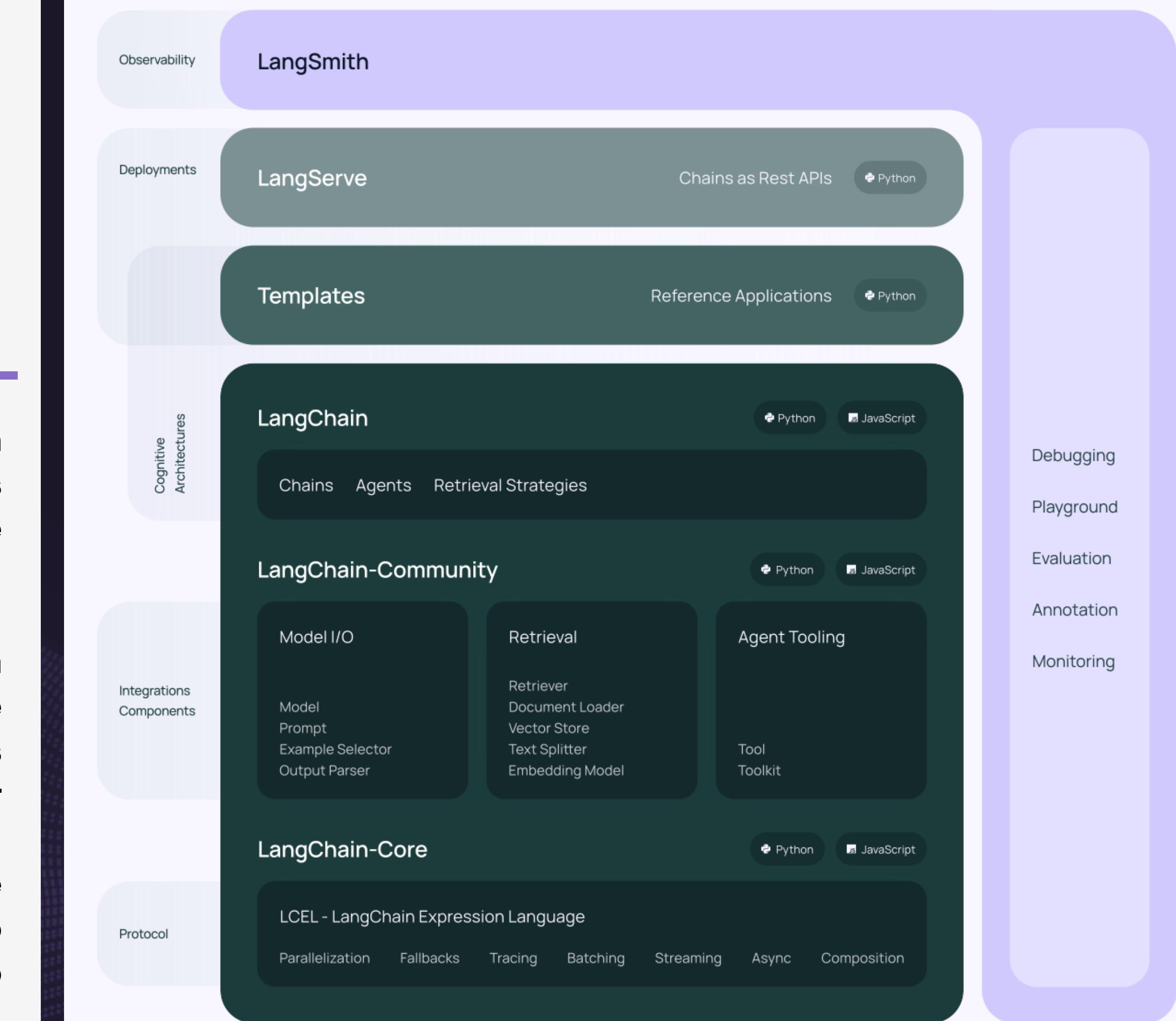




# LangChain

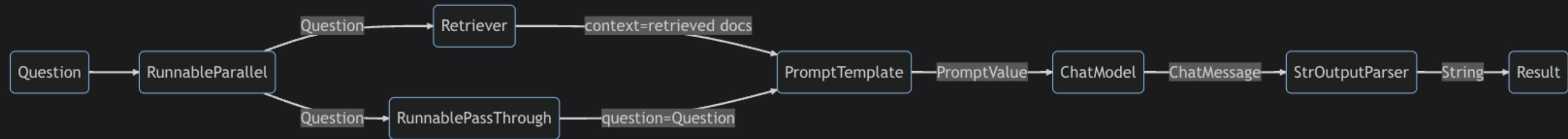
LangChain é um framework para desenvolvimento de aplicativos baseados em modelos de linguagem. Ele permite aplicativos que:

- São **context-aware**: conectam um modelo de linguagem a fontes de contexto (instruções imediatas, poucos exemplos, conteúdo para fundamentar sua resposta, etc.)
- Reason: confiar em um modelo de linguagem para raciocinar (sobre como responder com base no contexto fornecido, quais ações tomar, etc.)



# Porque usar LangChain Expression Language

# Basic RAG



# AI Agentic Workflows

**WHAT'S NEXT FOR AI AGENTIC WORKFLOWS**  
**FT. ANDREW NG OF AI FUND**

# Challenge 1

Criar um pipeline que receberá um input/pesquisa. Este pipeline precisa realizar uma pesquisa na web e resumir o assunto pesquisado e finalmente gerar um arquivo TXT.

## Challenge 2

Criar um agente para Q&A com RAG. Faça o embedding de um documento PDF no ChromaDB, use o banco como sua fonte de informações para o RAG. O agente dever ser conversacional e capaz de buscar as informações quando necessário.

## Challenge 3

Criar um agente que receberá um consulta a banco de dados utilizando linguagem natural. Esse agente deve ser capaz de manter uma conversa natural com usuário e responder perguntas consecutivas sobre o mesmo tópico.

Exemplos de perguntas consecutivas:

- Qual track foi mais vendida?
- Busque os dados da invoice desta track.
- Quem é o autor?



# MARCOS NATÃ



lambdaly.ai

[lambdaly.ai](https://lambdaly.ai)



[@marcosnataqs](https://www.linkedin.com/in/marcosnataqs)



[@marcosnataqs](https://github.com/marcosnataqs)



[@marcosnataqs](https://twitter.com/marcosnataqs)



[marcos.nata@lambdaly.ai](mailto:marcos.nata@lambdaly.ai)



+55 62 99309-5562

