

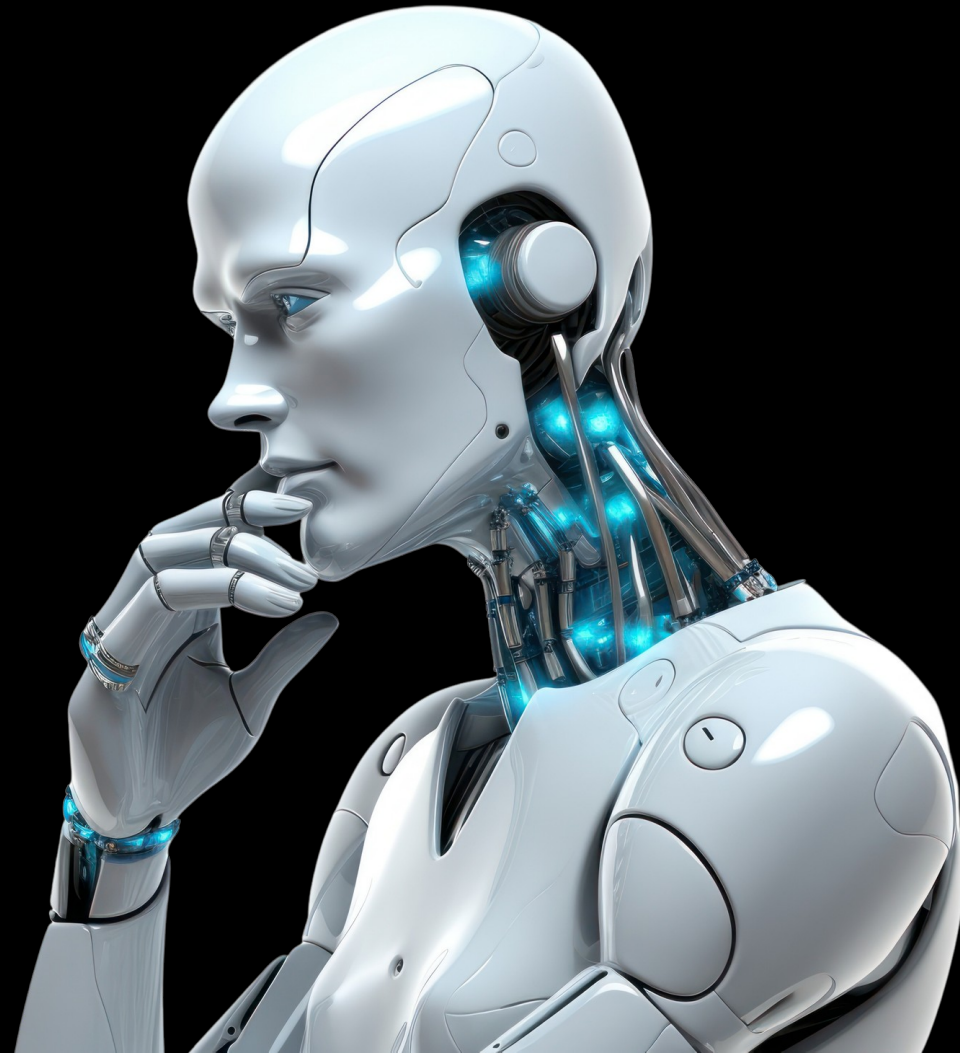
LLMs (2)

Features++ & challenges

Aissam Outchakoucht

a.outchakoucht@emsi-edu.ma

aissam.outchakoucht@gmail.com



3.5
2.0
0.2
.
.
.

Recap: Tokenization, EMbedding

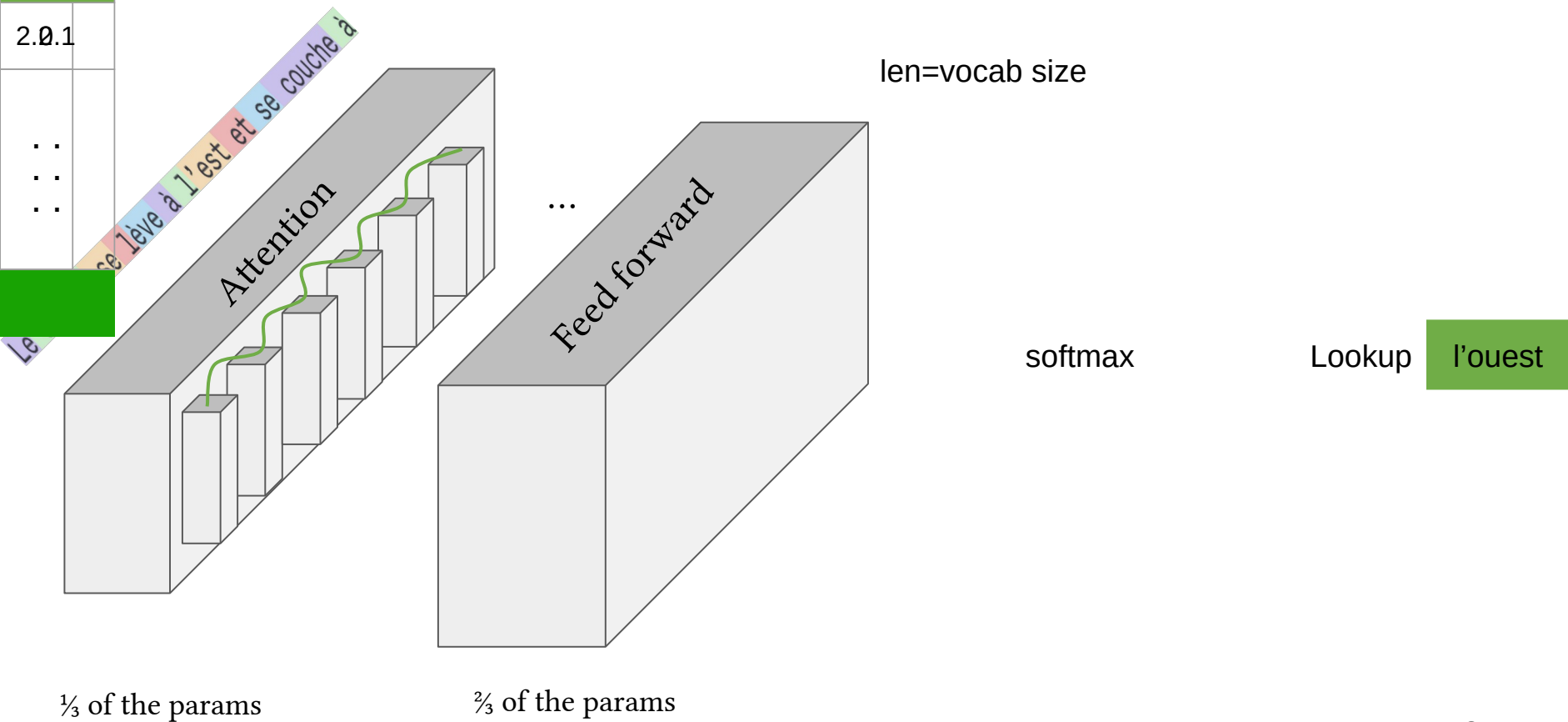
?

Le soleil se lève à l'est et se couche à

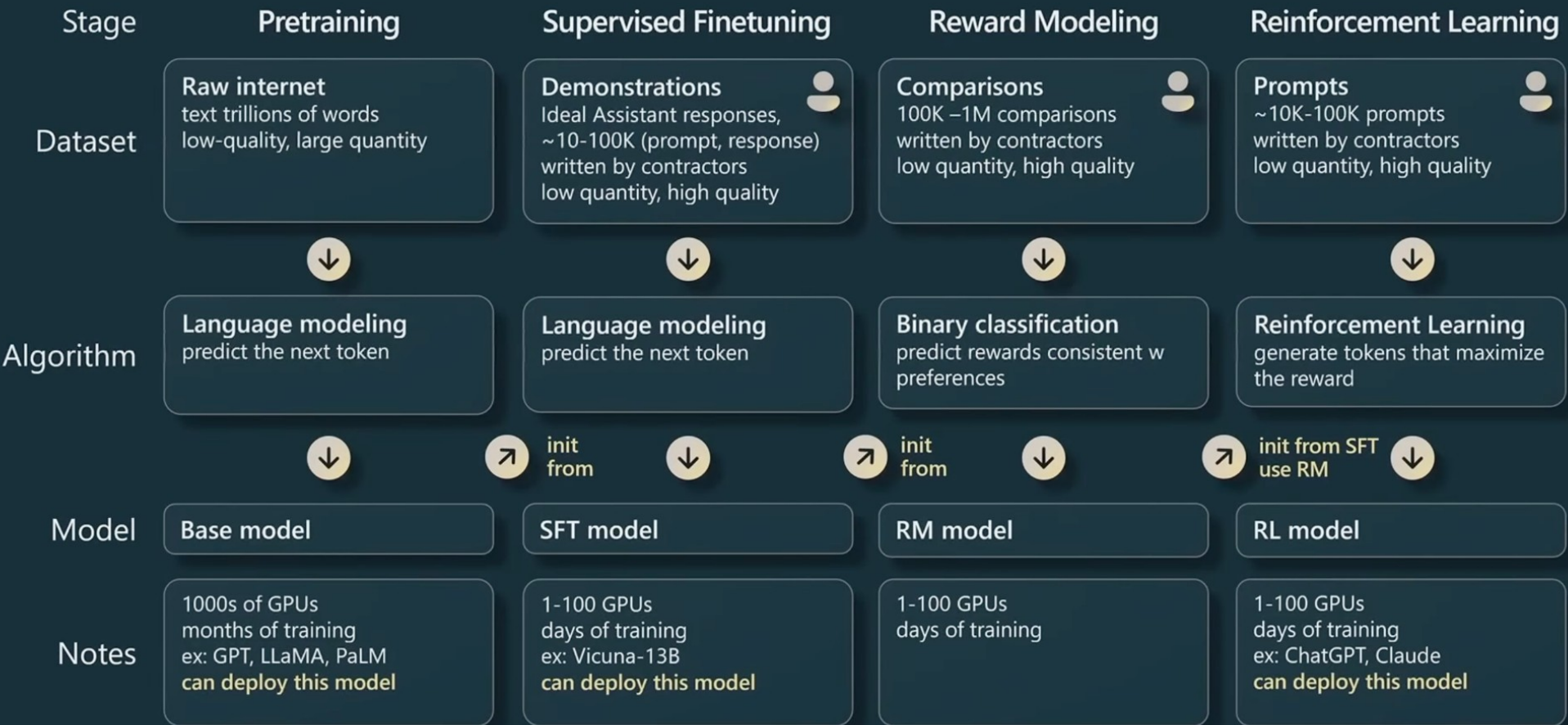
...

0.001
0.87
2.0.1
...
...
...

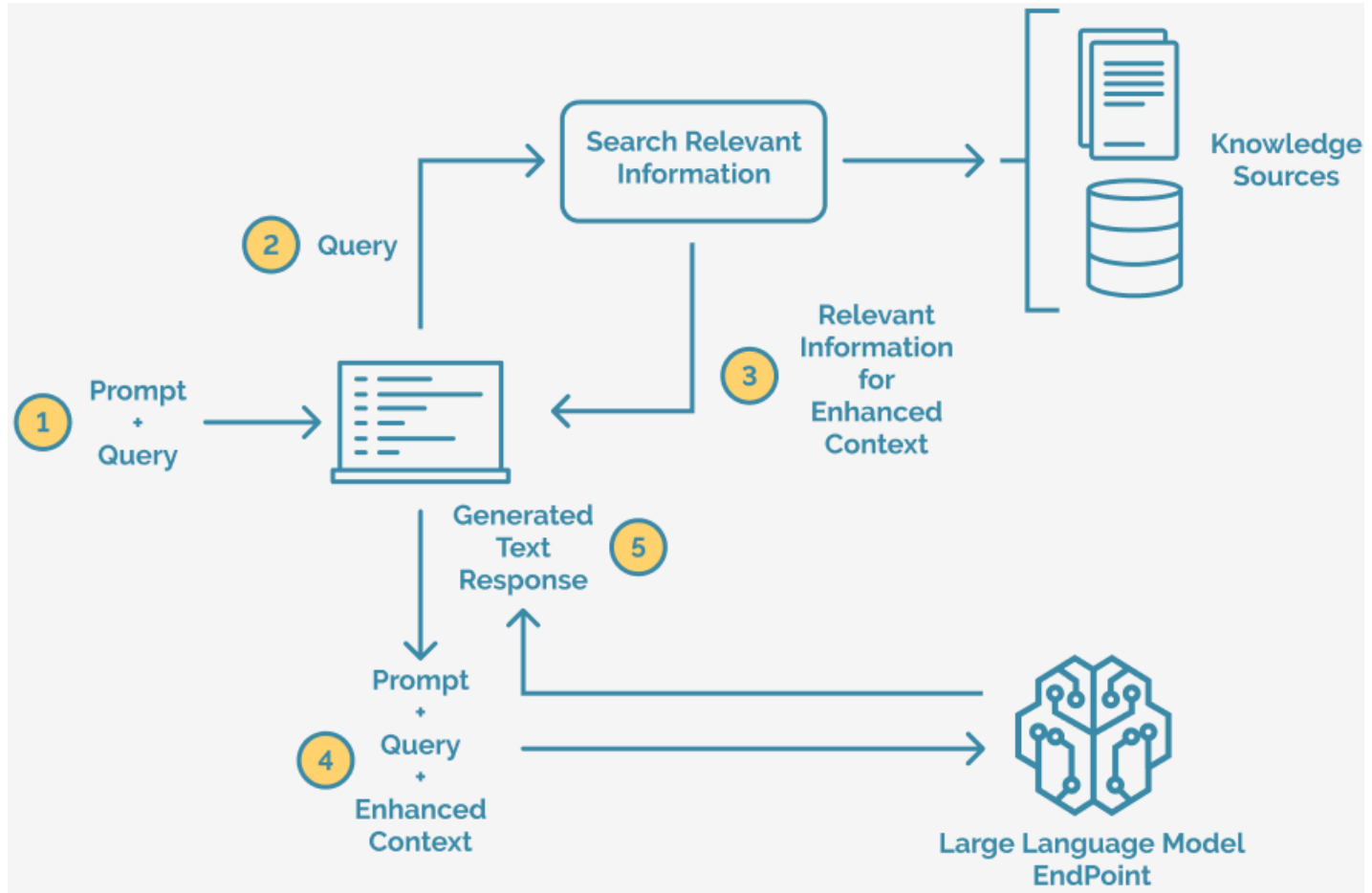
Recap: Transformers



GPT Assistant training pipeline



Retrieval augmented generation -RAG-



RAG

RAG (Retrieval-Augmented Generation) est une technique qui combine deux composants :

1. Récupération (Retrieval) : Recherche d'informations pertinentes dans une base de connaissances externe (par ex. une base documentaire).
1. Génération (Generation) : Utilisation d'un modèle génératif (comme GPT) pour formuler une réponse basée sur les informations récupérées.

RAG

Pourquoi utiliser RAG ?

1. Permet de répondre à des questions nécessitant des informations précises et actualisées.
1. Réduit la dépendance aux données statiques dans les modèles de langage.
1. Idéal pour des cas comme :
 - a. Questions-réponses contextuelles.
 - b. Génération de résumés basés sur des documents.
 - c. Recherche documentaire interactive.

Fine-tuning

Fine-tuning est le processus d'adaptation d'un modèle pré-entraîné à une tâche spécifique.

Pourquoi fine-tuner un modèle ?

- Le modèle pré-entraîné a appris des représentations générales (par ex. GPT, BERT).
- La tâche spécifique (par ex. classification de textes médicaux) nécessite une adaptation aux données particulières.

Fine-tuning

Create a fine-tuned model

Method

Specify the method to be used for fine-tuning.

Supervised



Base Model

gpt-4o-2024-08-06



Training data

Add a jsonl file to use for training. By providing the file, you confirm that you have the rights to use the data.

☒ Upload new ☐ Select existing



Upload a file or drag and drop here

(.jsonl)

Validation data

Add a jsonl file to use for validation metrics.

☐ Upload new ☐ Select existing ☒ None

Create a fine-tuned model

Add a custom suffix that will be appended to the output model name.

my-experiment

Seed

The seed controls the reproducibility of the job. Passing in the same seed and job parameters should produce the same results, but may differ in rare cases. If a seed is not specified, one will be generated for you.

Random

Configure hyperparameters

☒ Batch size ⓘ

auto

☒ Learning rate multiplier ⓘ

auto

In most cases, range of 0.0001- 10 is recommended

☒ Number of epochs ⓘ

auto

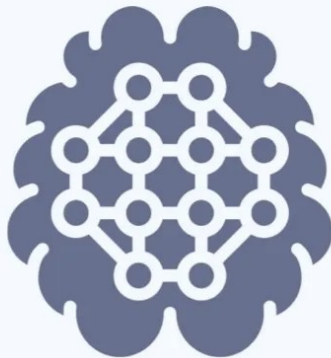
In most cases, range of 1- 10 is recommended

[Learn about fine-tuning ↗](#)

Cancel

Create

Function calling



Prompt

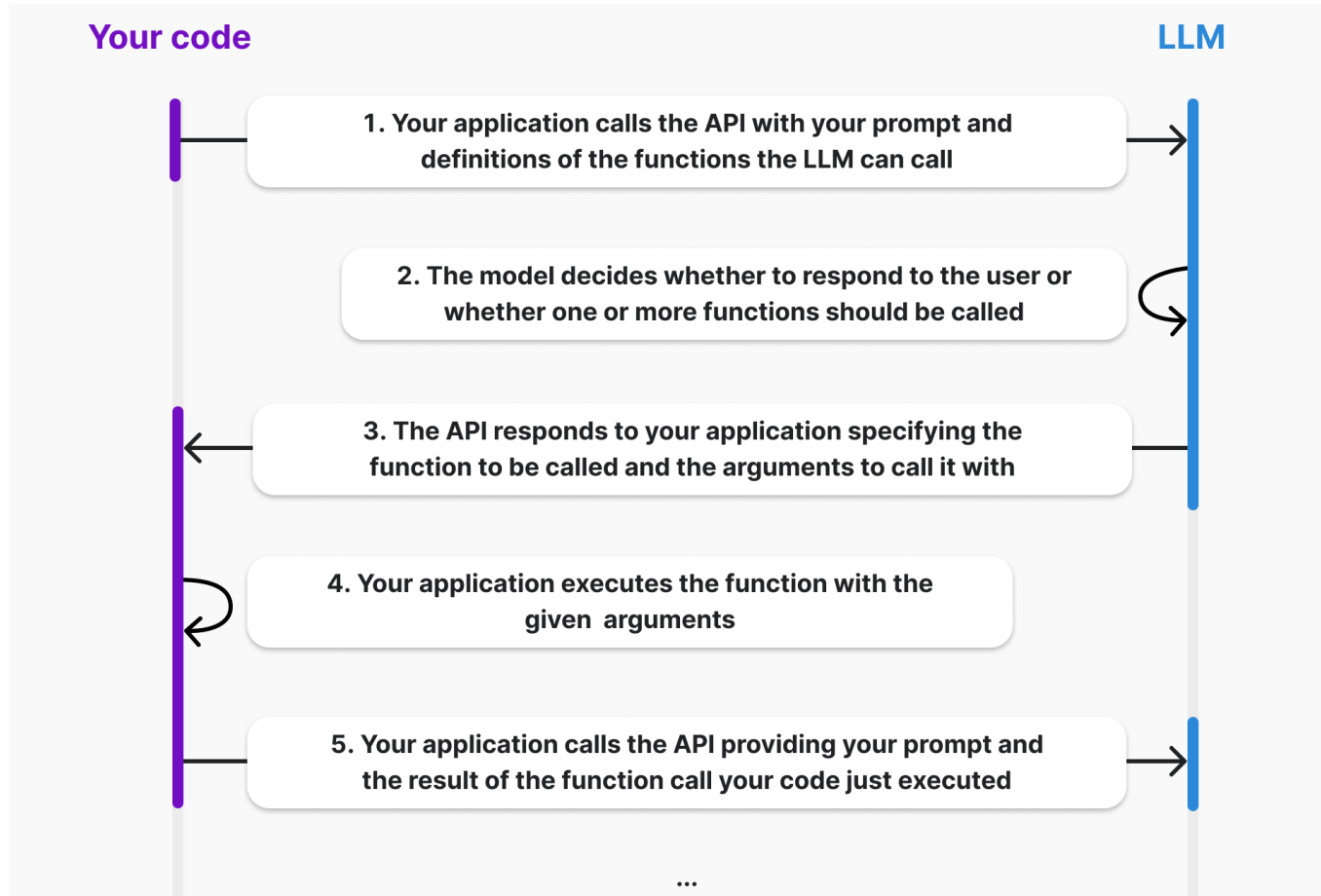
LLM

Functions

Function calling

Permet aux développeurs de connecter des modèles de langage à des données et des systèmes externes. Vous pouvez définir un ensemble de fonctions comme des outils auxquels le modèle a accès, et il peut les utiliser lorsque c'est pertinent en fonction de l'historique de la conversation. Vous pouvez ensuite exécuter ces fonctions côté application et renvoyer les résultats au modèle.

Function calling



Function calling

Fonctions dédiées :

- Elles peuvent réaliser des tâches précises (récupérer une information, effectuer un calcul, interagir avec une API...).

Accès conditionnel :

- Le modèle ne les appellera que si cela est pertinent dans le contexte de la conversation.

Exécution côté application :

- Les fonctions sont codées et exécutées en dehors du modèle.
- Les résultats sont ensuite renvoyés au modèle pour continuer la génération de réponses.

Exemple d'utilisation :

- Vérifier la disponibilité d'un produit dans un inventaire.
- Récupérer la météo via une API.
- Envoyer un e-mail ou un message.

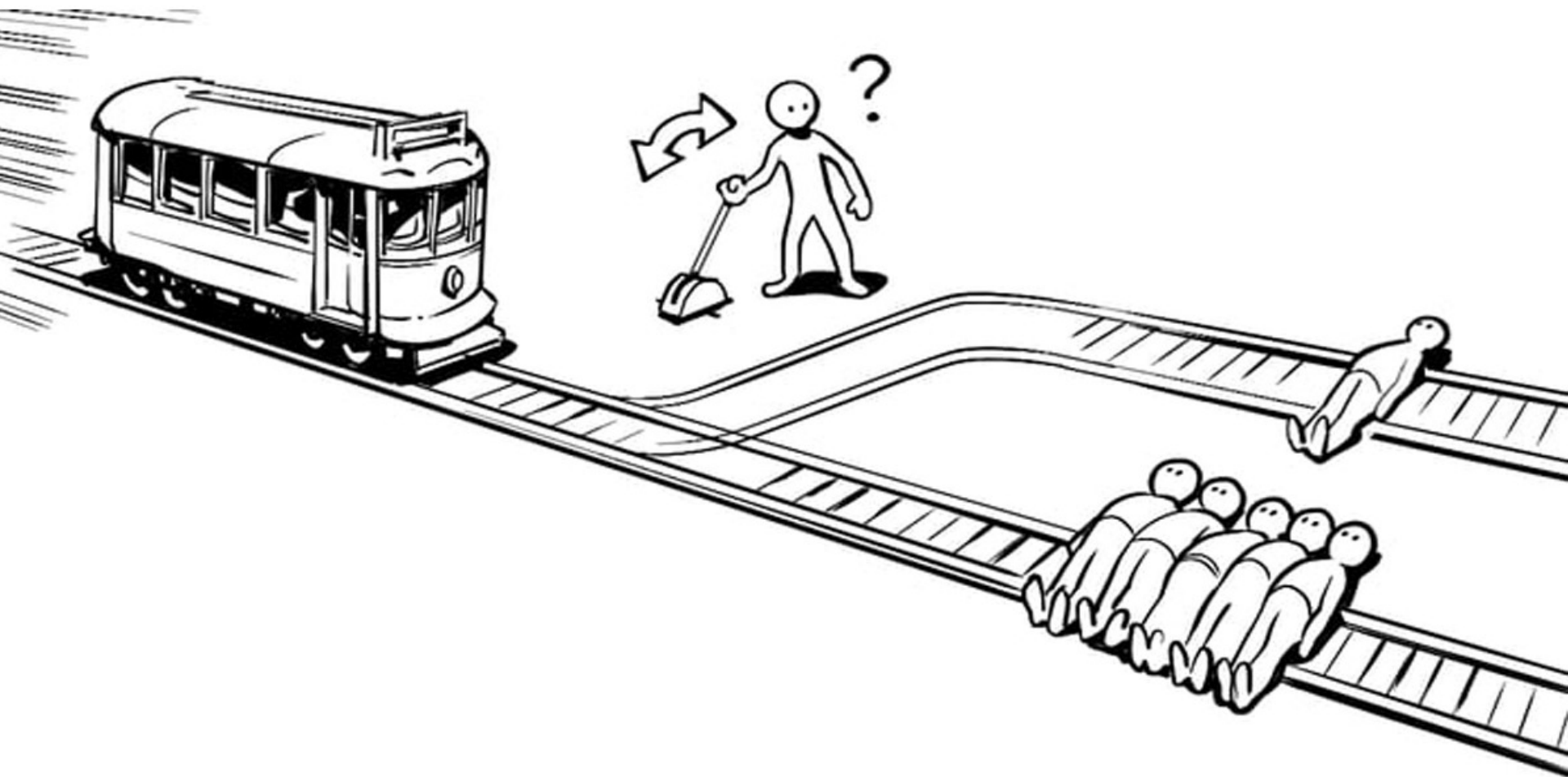
Function calling

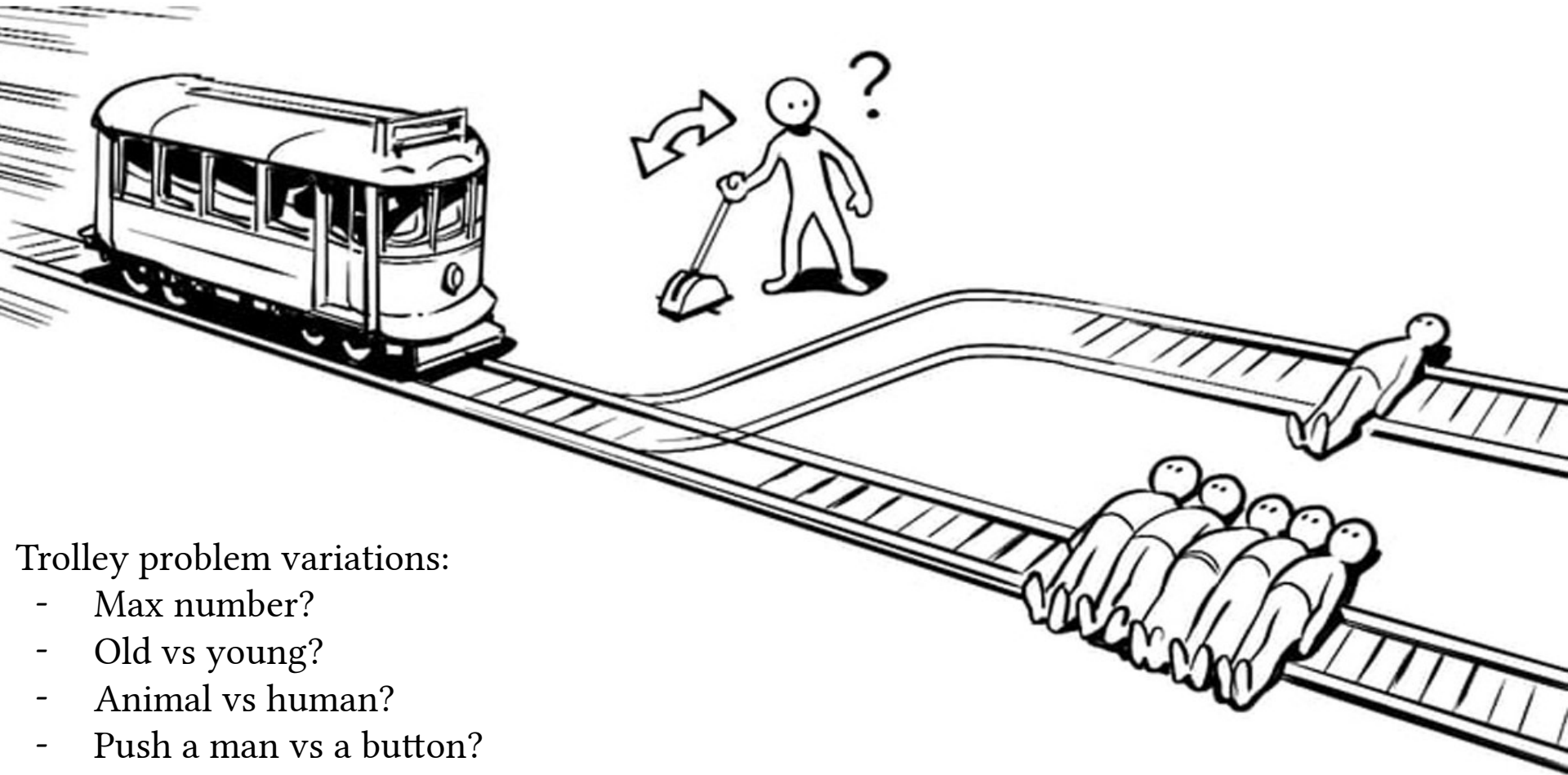
```
1  from openai import OpenAI
2
3  client = OpenAI()
4
5  tools = [
6      {
7          "type": "function",
8          "function": {
9              "name": "get_weather",
10             "parameters": {
11                 "type": "object",
12                 "properties": {
13                     "location": {"type": "string"}
14                 },
15             },
16         },
17     ]
18 ]
19
20 completion = client.chat.completions.create(
21     model="gpt-4o",
22     messages=[{"role": "user", "content": "What's the weather like in Paris today?"}],
23     tools=tools,
24 )
25
26 print(completion.choices[0].message.tool_calls)
```

Function calling

```
1 from openai import OpenAI
2
3 client = OpenAI()
4
5 tools = [
6     {
7         "type": "function",
8         "function": {
9             "name": "get_weather"
10            "parameters": {
11                "type": "object",
12                "properties": {
13                    "location": {"type": "string"}
14                },
15            },
16        },
17    ]
18 ]
19
20 completion = client.chat.completions.create(
21     model="gpt-4o",
22     messages=[{"role": "user", "content": "What's the weather like in Paris today?"}],
23     tools=tools,
24 )
25
26 print(completion.choices[0].message.tool_calls)
```

```
1 [
2     {
3         "id": "call_12345xyz",
4         "type": "function",
5         "function": { "name": "get_weather", "arguments": "{ 'location': 'Paris' }" }
6     }
7 ]
```





Trolley problem variations:

- Max number?
- Old vs young?
- Animal vs human?
- Push a man vs a button?
- Medical setup?

Adversarial attacks on nns

Les réseaux profonds obtiennent de très bons résultats sur des données légitimes MAIS

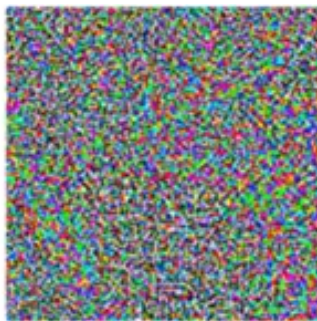
Il est extrêmement facile pour les “adversaires” (attaquants) de les tromper (surtout si ces derniers ont des connaissances à propos le modèle -boîte blanche-)



“panda”

57.7% confidence

+ .007 ×



noise

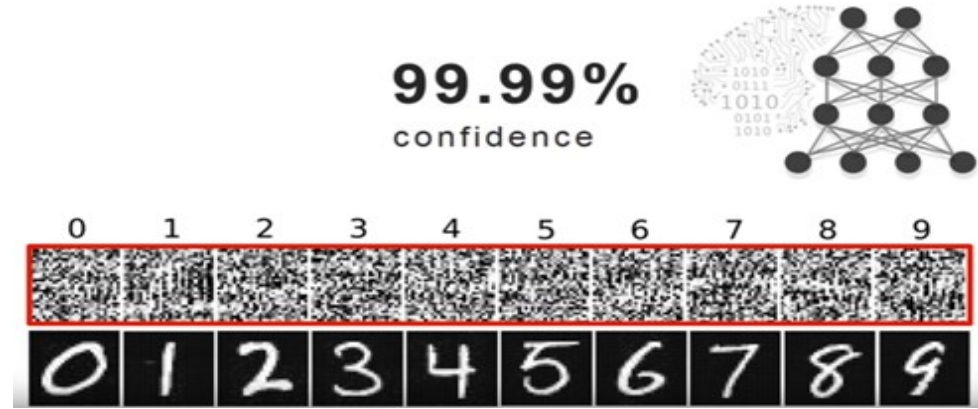
=



“gibbon”

99.3% confidence

Adversarial attacks on nns



Do what I mean, not what I say

- “Get my grandma out of the burning house”
- “Help me lose weight quickly.”
- “Keep me safe”
- ...