BOOTCAMP

Python Aplicado à Análise de Dados e IA para o Setor Elétrico Brasileiro

IEEE | Universidade Federal do Rio Grande do Norte IEEE Student Branch | IEEE PES Power & Energy Society

C2AI | CAECOMP | IEEE COMPUTER SOCIETY

Introdução a Agentes com LLMs

MATHEUS GOMES DINIZ ANDRADE
matheus.diniz.122@ufrn.edu.br

WHO I AM

Matheus Andrade

Bachelor of Science and Technology - **UFRN**

Computer Engineer - **UFRN**

Master's Candidate - **PPgEEC/UFRN**

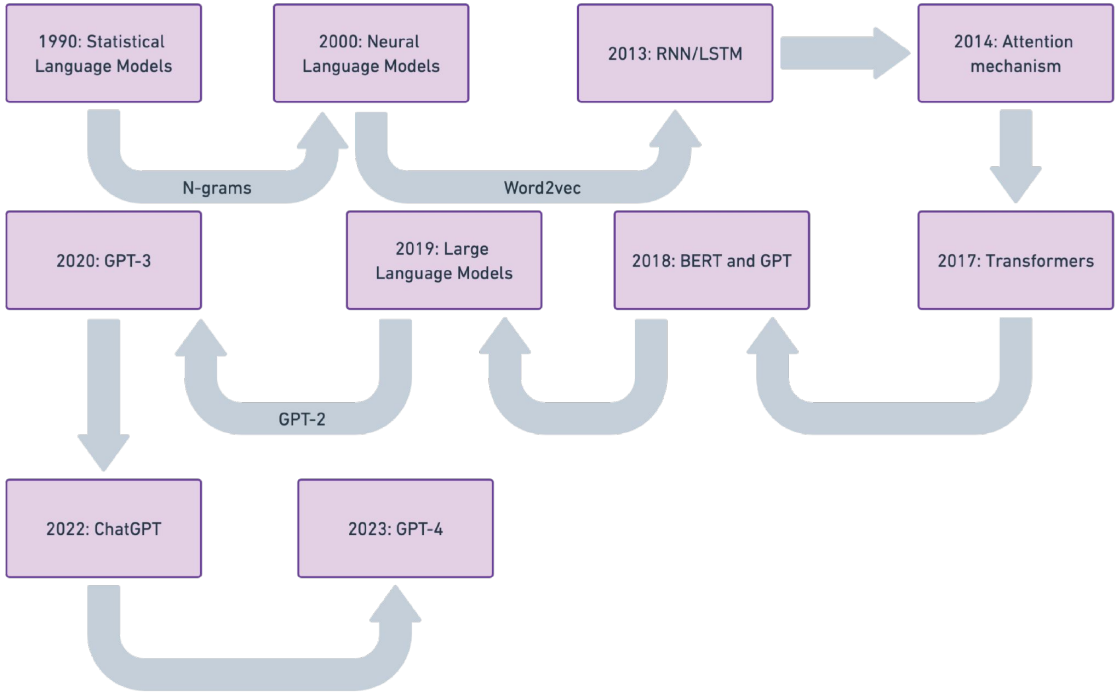
Member of the **Conect2ai** Research Group | Projects: **CNPq**, **Rota2030** and **PlaforEDU**

INTRODUCTION

WHAT ARE LLMS?

INTRODUCTION

A BRIEF HISTORY



WHAT ARE LLMS?

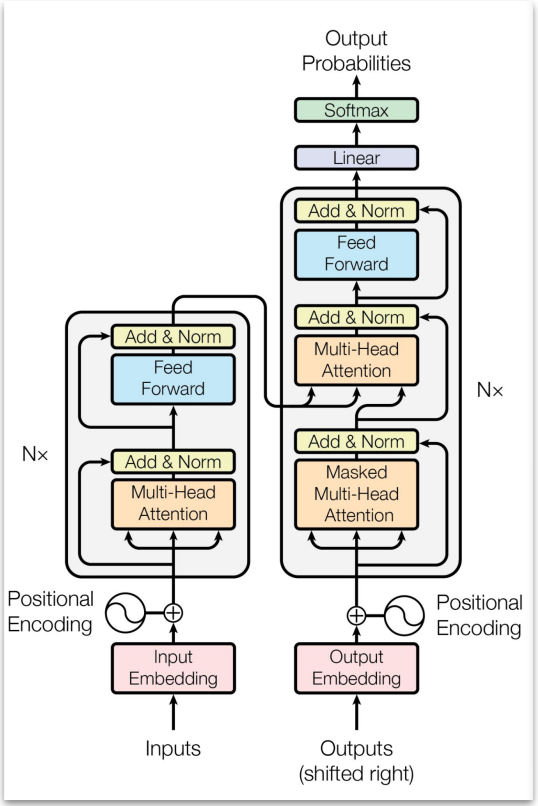
Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

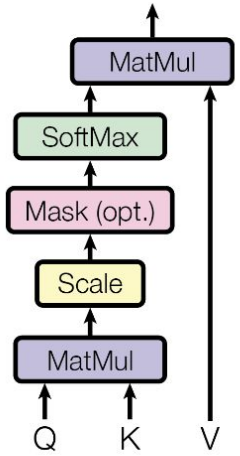
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and



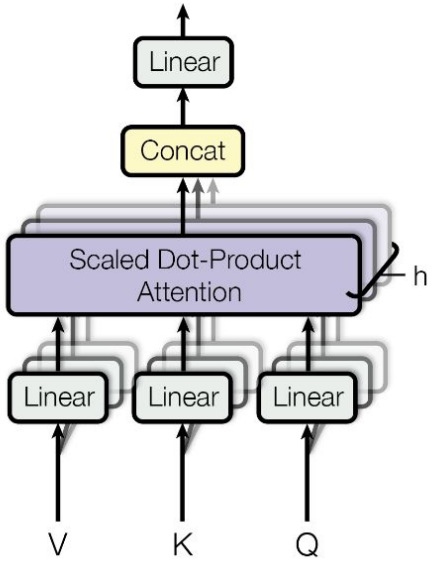
WHAT ARE LLMS?

Scaled Dot-Product Attention



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

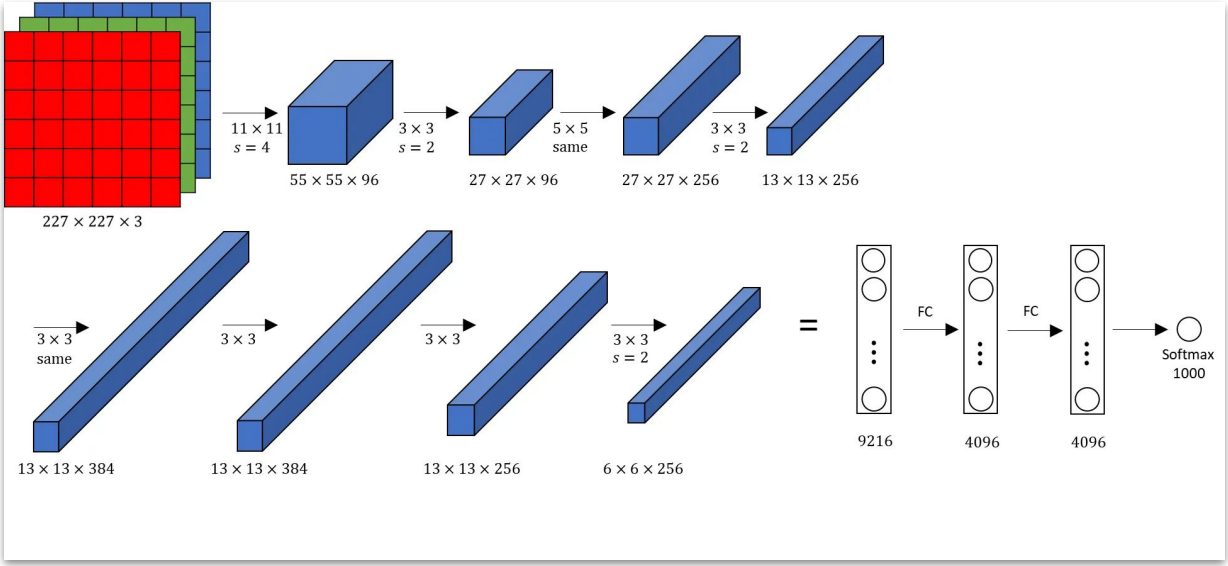
Multi-Head Attention



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

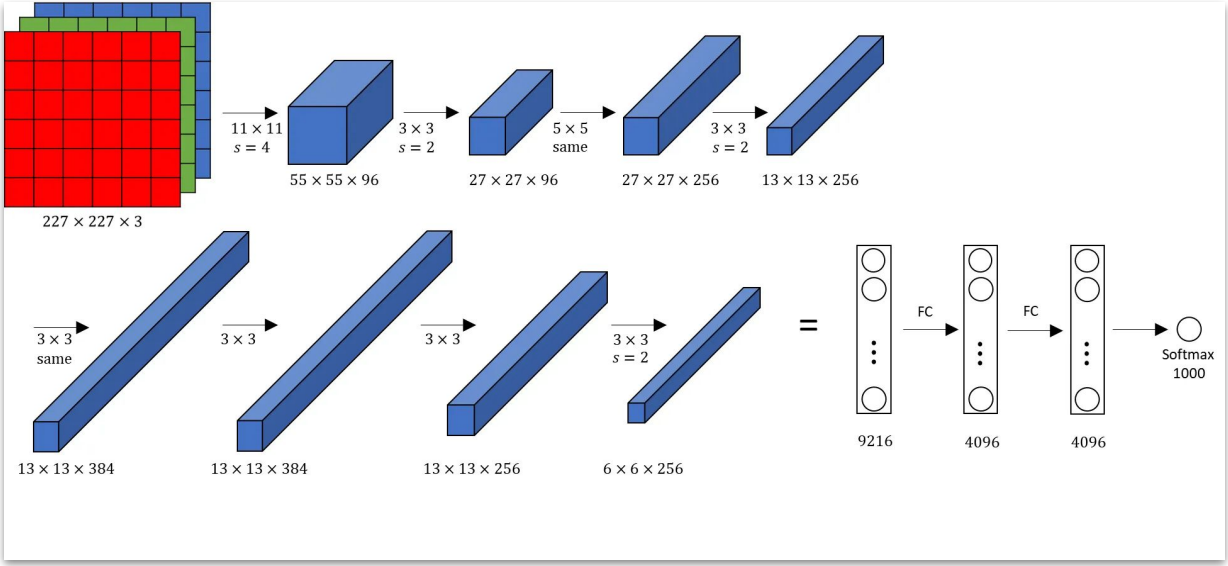
INTRODUCTION

WHAT ARE LLMS?



INTRODUCTION

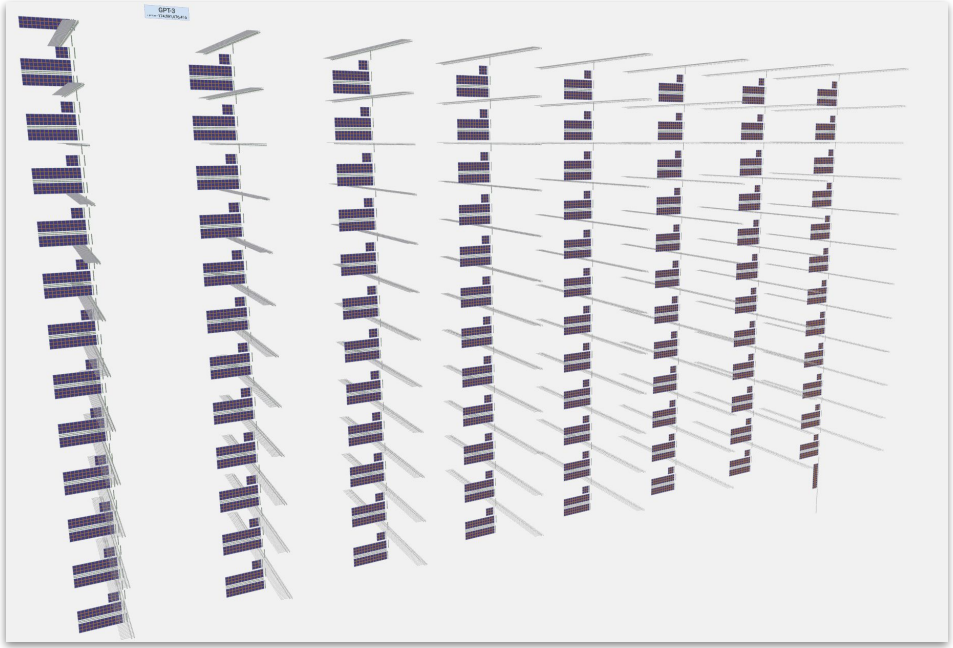
WHAT ARE LLMS?



60 Milhões

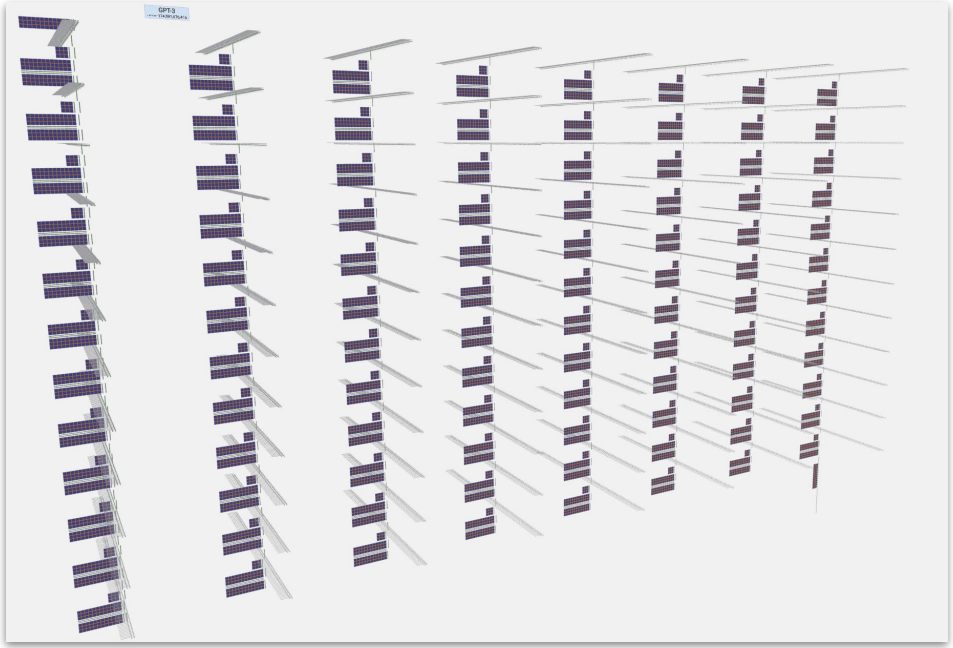
INTRODUCTION

WHAT ARE LLMS?

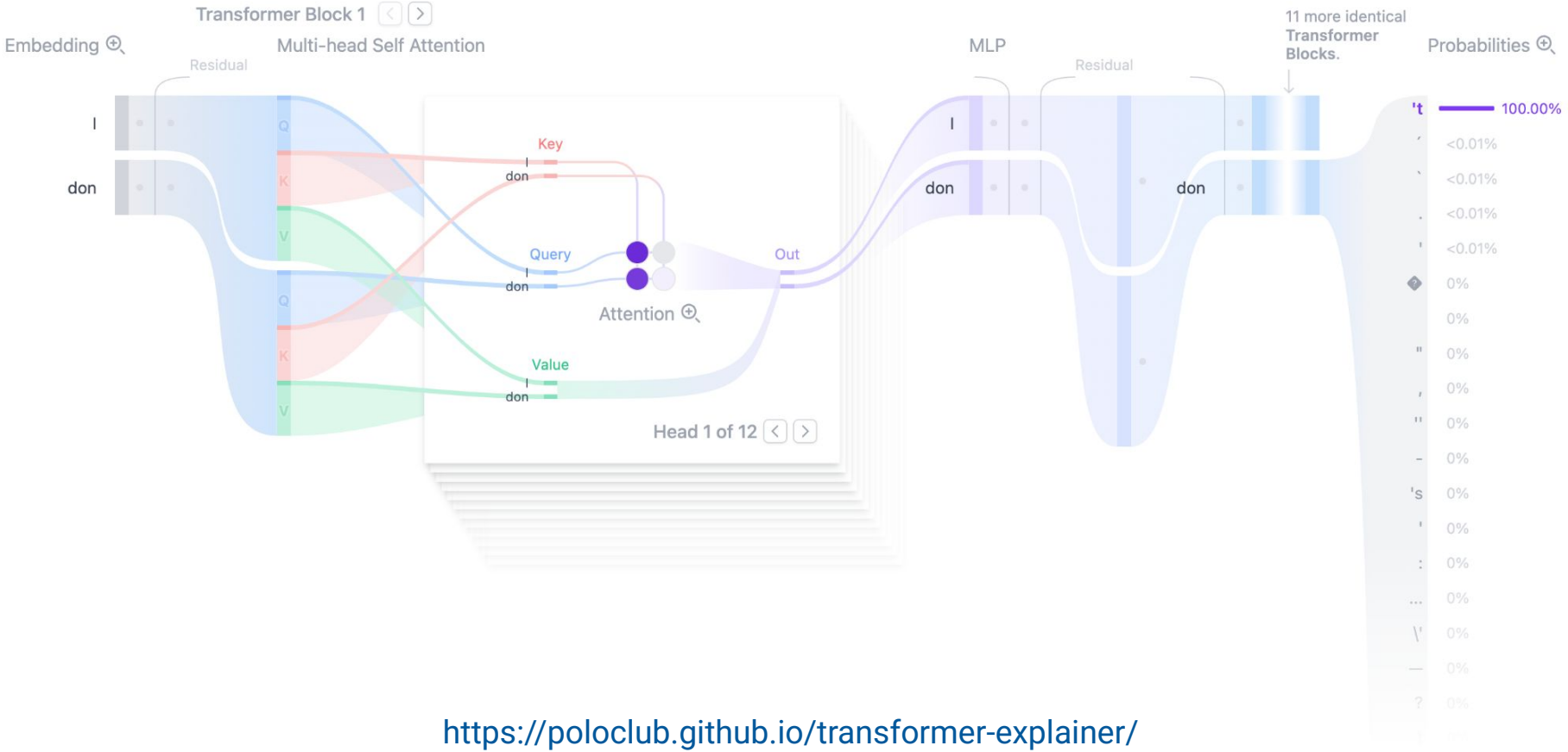


INTRODUCTION

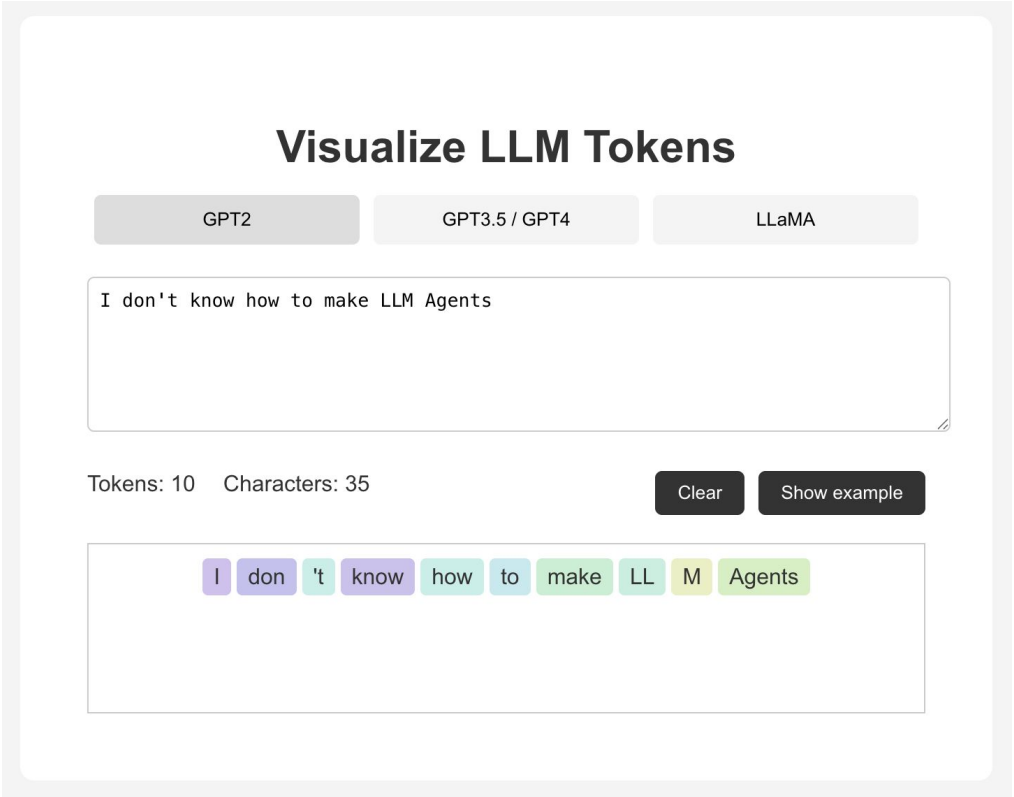
WHAT ARE LLMS?



175 Bilhões



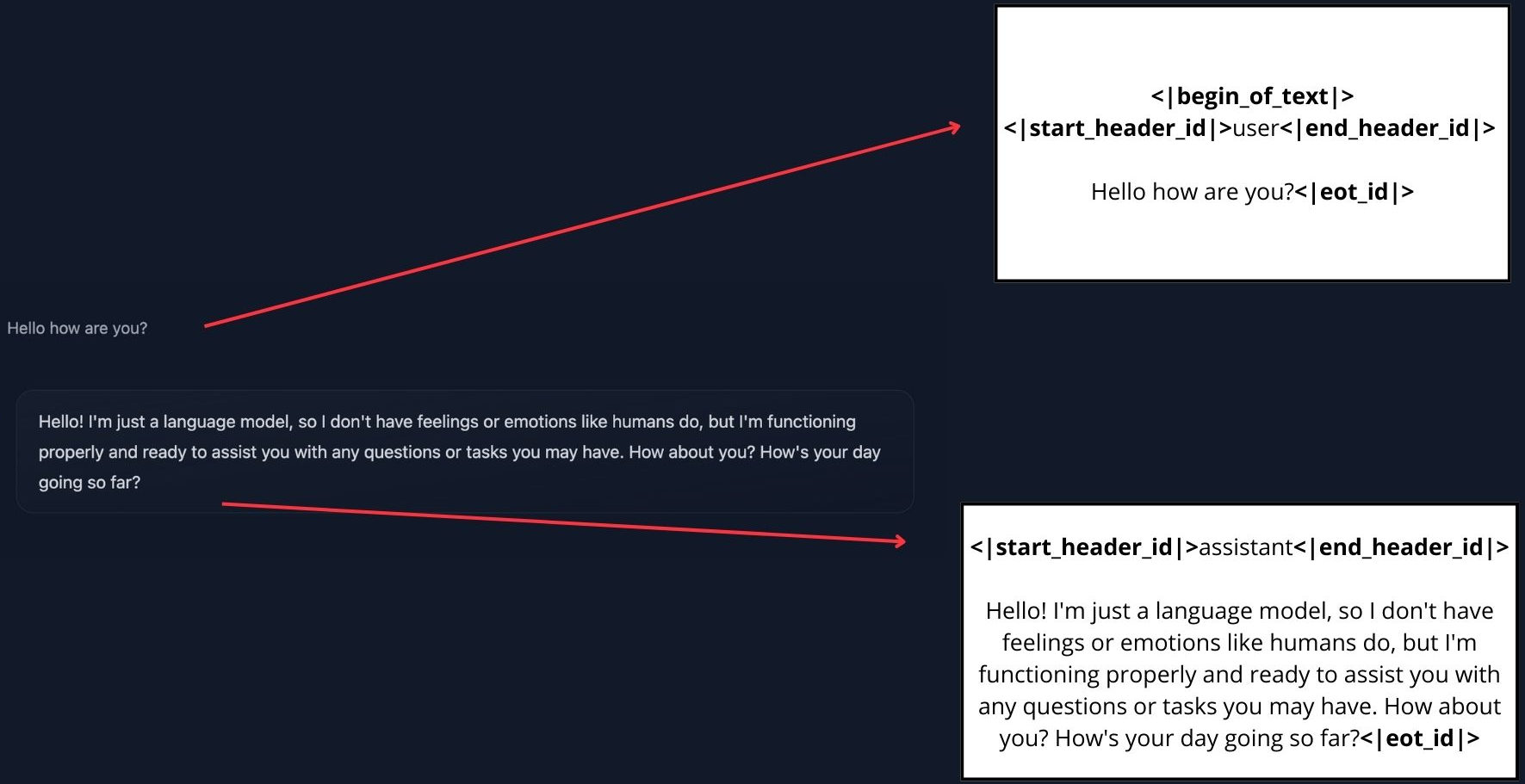
https://poloclub.github.io/transformer-explainer/



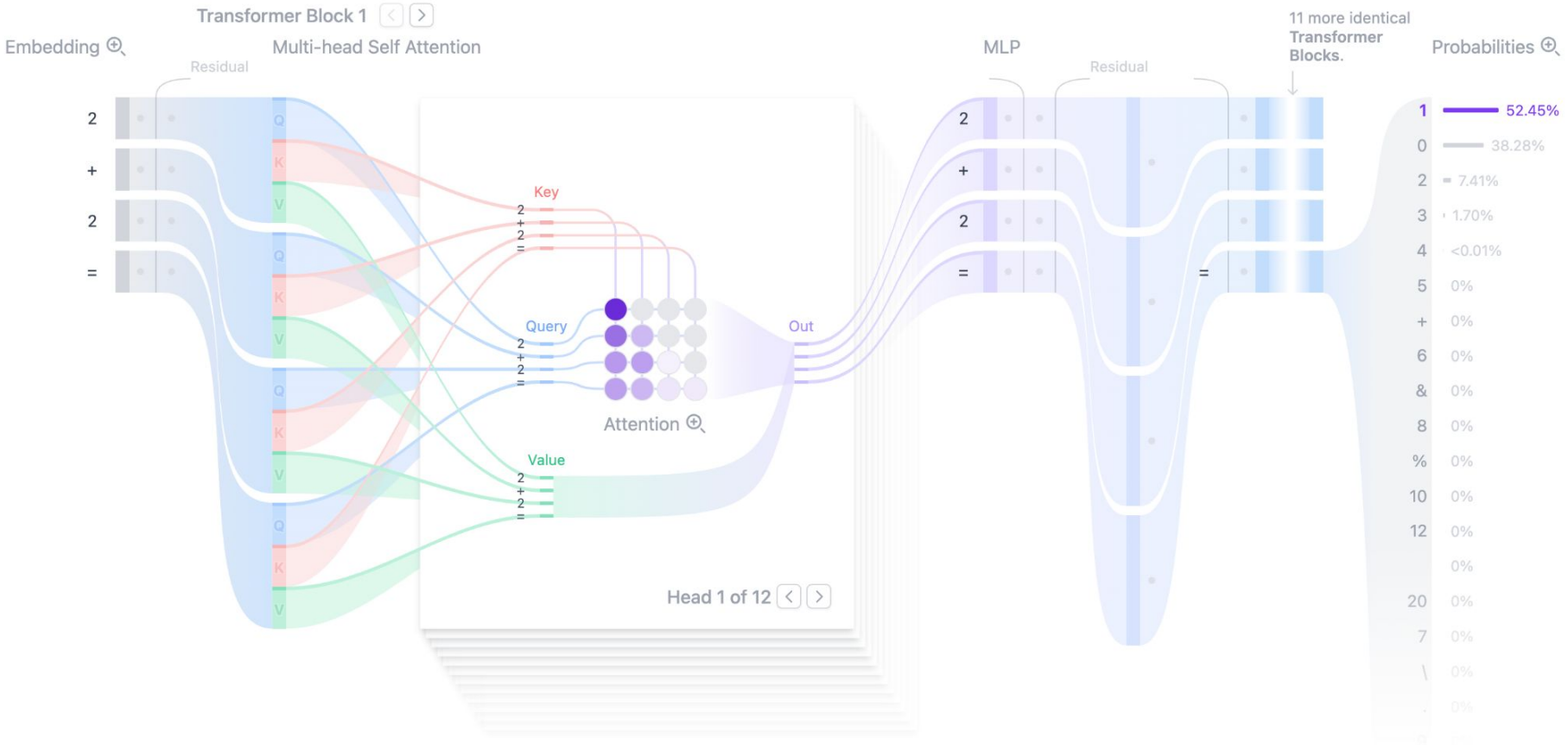
https://tokenvisualizer.netlify.app/

SPECIAL TOKENS

```
▶Uncaught Error: The text contains a special token that is
not allowed: <|endoftext|>
    at App.tsx:35:5
    at App.tsx:75:18
  Show ignore-listed frames
```



source: Hugging Face


Transformer Block 1
Embedding
Multi-head Self Attention
Residual
MLP
Residual
11 more identical Transformer Blocks.
Probabilities
Key
Query
Value
Attention
Out
Head 1 of 12
52.45%
38.28%
7.41%
1.70%
<0.01%


Transformer Block 1
Embedding
Multi-head Self Attention
Residual
MLP
Residual
11 more identical Transformer Blocks.
Probabilities
Key
Query
Value
Out
of 12
1 52.45%
0 38.28%
2 7.41%
3 1.70%
4 <0.01%

LLM AGENT

WHAT IS AN AGENT?

What is 2 + 2?

sum_tool

subtract_tool

division_tool

multiplication_tool

WHAT IS AN AGENT?

What is 2 + 2?

sum_tool

subtract_tool

division_tool

multiplication_tool

WHAT IS AN AGENT?

What is 2 + 2?

sum_tool(2, 2)

WHAT IS AN AGENT?

What is 2 + 2?

PILLARS

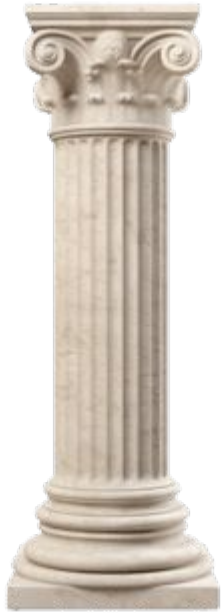
PILLARS

THOUGHT

PILLARS

THOUGHT

ACTION



INTRODUCTION

PILLARS

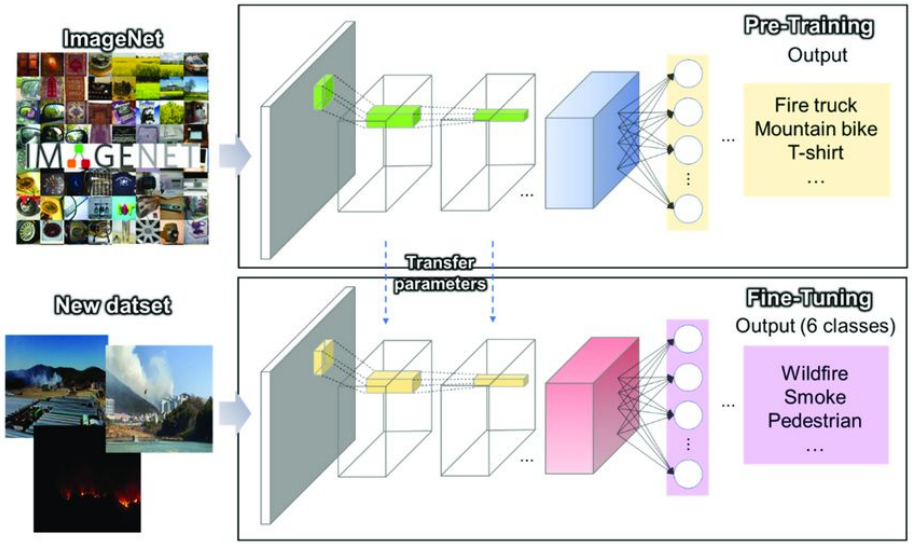
THOUGHT

ACTION

OBSERVATION

HOW AGENTS ARE TRAINED?

FINE-TUNING


ImageNet
IMAGENET
Pre-Training
Output
Fire truck
Mountain bike
T-shirt
...
Transfer
parameters
New datset
Fine-Tuning
Output (6 classes)
Wildfire
Smoke
Pedestrian
...

HOW AGENTS ARE TRAINED?

HOW AGENTS ARE TRAINED?

```
{
"content": "You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags.You may call one or more
functions to assist with the user query. Don't make assumptions about what values to plug into functions.Here are the available tools:<tools> [{'type':
'function', 'function': {'name': 'get_stock_price', 'description': 'Get the current stock price of a company', 'parameters': {'type': 'object',
'properties': {'company': {'type': 'string', 'description': 'The name of the company'}}, 'required': ['company']}}}, {'type': 'function', 'function':
{'name': 'get_movie_details', 'description': 'Get details about a movie', 'parameters': {'type': 'object', 'properties': {'title': {'type': 'string',
'description': 'The title of the movie'}}, 'required': ['title']}}}] </tools>Use the following pydantic model json schema for each tool call you will
make: {'title': 'FunctionCall', 'type': 'object', 'properties': {'arguments': {'title': 'Arguments', 'type': 'object'}, 'name': {'title': 'Name', 'type':
'string'}}, 'required': ['arguments', 'name']}For each function call return a json object with function name and arguments within <tool_call></tool_call>
XML tags as follows:\n<tool_call>\n{tool_call}\n</tool_call>",
"role": "system"
},
```

HOW AGENTS ARE TRAINED?

```
{
"content": "You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags.You may call one or more
functions to assist with the user query. Don't make assumptions about what values to plug into functions.Here are the available tools:<tools> [{'type':
'function', 'function': {'name': 'get_stock_price', 'description': 'Get the current stock price of a company', 'parameters': {'type': 'object',
'properties': {'company': {'type': 'string', 'description': 'The name of the company'}}, 'required': ['company']}}}, {'type': 'function', 'function':
{'name': 'get_movie_details', 'description': 'Get details about a movie', 'parameters': {'type': 'object', 'properties': {'title': {'type': 'string',
'description': 'The title of the movie'}}, 'required': ['title']}}}] </tools>Use the following pydantic model json schema for each tool call you will
make: {'title': 'FunctionCall', 'type': 'object', 'properties': {'arguments': {'title': 'Arguments', 'type': 'object'}, 'name': {'title': 'Name', 'type':
'string'}}, 'required': ['arguments', 'name']}For each function call return a json object with function name and arguments within <tool_call></tool_call>
XML tags as follows:\n<tool_call>\n{tool_call}\n</tool_call>",
"role": "system"
},
{
"content": "Hi, can you tell me the current stock price of Apple?",
"role": "human"
},
```

HOW AGENTS ARE TRAINED?

```
{
"content": "You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags.You may call one or more
functions to assist with the user query. Don't make assumptions about what values to plug into functions.Here are the available tools:<tools> [{'type':
'function', 'function': {'name': 'get_stock_price', 'description': 'Get the current stock price of a company', 'parameters': {'type': 'object',
'properties': {'company': {'type': 'string', 'description': 'The name of the company'}}, 'required': ['company']}}}, {'type': 'function', 'function':
{'name': 'get_movie_details', 'description': 'Get details about a movie', 'parameters': {'type': 'object', 'properties': {'title': {'type': 'string',
'description': 'The title of the movie'}}, 'required': ['title']}}}] </tools>Use the following pydantic model json schema for each tool call you will
make: {'title': 'FunctionCall', 'type': 'object', 'properties': {'arguments': {'title': 'Arguments', 'type': 'object'}, 'name': {'title': 'Name', 'type':
'string'}}, 'required': ['arguments', 'name']}For each function call return a json object with function name and arguments within <tool_call></tool_call>
XML tags as follows:\n<tool_call>\n{tool_call}\n</tool_call>",
"role": "system"
},
{
"content": "Hi, can you tell me the current stock price of Apple?",
"role": "human"
},
{
"content": "<think>Alright, the user asked for the current stock price of Apple. Looking at the tools available, there's a function called
get_stock_price which exactly fits this request. I'll need to call this function with the parameter company set to 'Apple' to retrieve the information
the user is asking for. This seems like the most straightforward and appropriate action to take.\n</think><tool_call>\n{'name': 'get_stock_price',
'arguments': {'company': 'Apple'}}\n</tool_call>",
"role": "model"
},
```

HOW AGENTS ARE TRAINED?


```
{
"content": "You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags.You may call one or more
functions to assist with the user query. Don't make assumptions about what values to plug into functions.Here are the available tools:<tools> [{'type':
'function', 'function': {'name': 'get_stock_price', 'description': 'Get the current stock price of a company', 'parameters': {'type': 'object',
'properties': {'company': {'type': 'string', 'description': 'The name of the company'}}, 'required': ['company']}}}, {'type': 'function', 'function':
{'name': 'get_movie_details', 'description': 'Get details about a movie', 'parameters': {'type': 'object', 'properties': {'title': {'type': 'string',
'description': 'The title of the movie'}}, 'required': ['title']}}}] </tools>Use the following pydantic model json schema for each tool call you will
make: {'title': 'FunctionCall', 'type': 'object', 'properties': {'arguments': {'title': 'Arguments', 'type': 'object'}, 'name': {'title': 'Name', 'type':
'string'}}, 'required': ['arguments', 'name']}For each function call return a json object with function name and arguments within <tool_call></tool_call>
XML tags as follows:\n<tool_call>\n{tool_call}\n</tool_call>",
"role": "system"
},
{
"content": "Hi, can you tell me the current stock price of Apple?",
"role": "human"
},
{
"content": "<think>Alright, the user asked for the current stock price of Apple. Looking at the tools available, there's a function called
get_stock_price which exactly fits this request. I'll need to call this function with the parameter company set to 'Apple' to retrieve the information
the user is asking for. This seems like the most straightforward and appropriate action to take.\n</think><tool_call>\n{'name': 'get_stock_price',
'arguments': {'company': 'Apple'}}\n</tool_call>",
"role": "model"
},
{
"content": "<tool_response>\n{'stock_price': '$150.75'}\n</tool_response>",
"role": "tool"
},
```

HOW AGENTS ARE TRAINED?

```
{
"content": "You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags.You may call one or more
functions to assist with the user query. Don't make assumptions about what values to plug into functions.Here are the available tools:<tools> [{'type':
'function', 'function': {'name': 'get_stock_price', 'description': 'Get the current stock price of a company', 'parameters': {'type': 'object',
'properties': {'company': {'type': 'string', 'description': 'The name of the company'}}, 'required': ['company']}}}, {'type': 'function', 'function':
{'name': 'get_movie_details', 'description': 'Get details about a movie', 'parameters': {'type': 'object', 'properties': {'title': {'type': 'string',
'description': 'The title of the movie'}}, 'required': ['title']}}}] </tools>Use the following pydantic model json schema for each tool call you will
make: {'title': 'FunctionCall', 'type': 'object', 'properties': {'arguments': {'title': 'Arguments', 'type': 'object'}, 'name': {'title': 'Name', 'type':
'string'}}, 'required': ['arguments', 'name']}For each function call return a json object with function name and arguments within <tool_call></tool_call>
XML tags as follows:\n<tool_call>\n{tool_call}\n</tool_call>",
"role": "system"
},
{
"content": "Hi, can you tell me the current stock price of Apple?",
"role": "human"
},
{
"content": "<think>Alright, the user asked for the current stock price of Apple. Looking at the tools available, there's a function called
get_stock_price which exactly fits this request. I'll need to call this function with the parameter company set to 'Apple' to retrieve the information
the user is asking for. This seems like the most straightforward and appropriate action to take.\n</think><tool_call>\n{'name': 'get_stock_price',
'arguments': {'company': 'Apple'}}\n</tool_call>",
"role": "model"
},
{
"content": "<tool_response>\n{'stock_price': '$150.75'}\n</tool_response>",
"role": "tool"
},
{
"content": "The current stock price of Apple is $150.75.",
"role": "model"
},
```

HOW AGENTS ARE TRAINED?

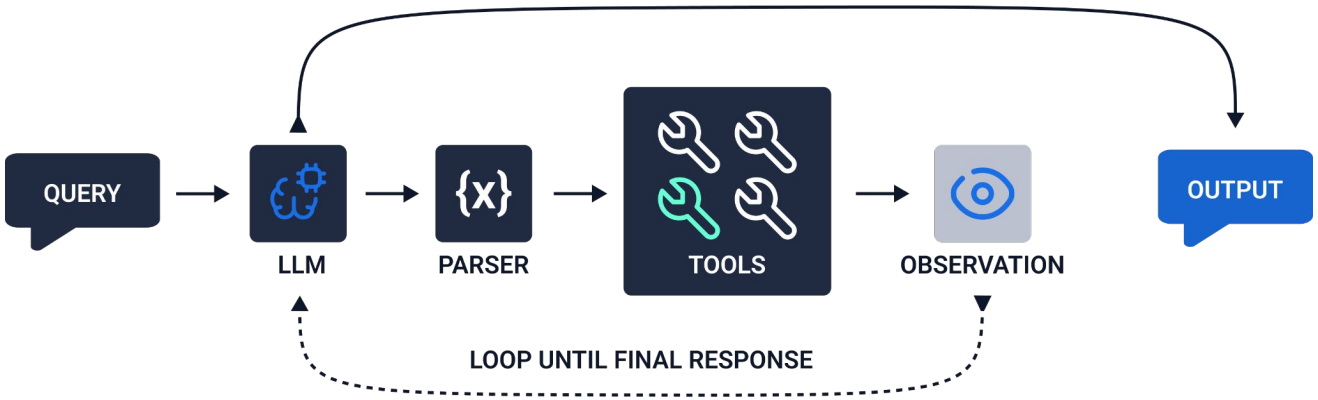

```
{
"content": "You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags.You may call one or more
functions to assist with the user query. Don't make assumptions about what values to plug into functions.Here are the available tools:<tools> [{'type':
'function', 'function': {'name': 'get_stock_price', 'description': 'Get the current stock price of a company', 'parameters': {'type': 'object',
'properties': {'company': {'type': 'string', 'description': 'The name of the company'}}, 'required': ['company']}}}, {'type': 'function', 'function':
{'name': 'get_movie_details', 'description': 'Get details about a movie', 'parameters': {'type': 'object', 'properties': {'title': {'type': 'string',
'description': 'The title of the movie'}}, 'required': ['title']}}}] </tools>Use the following pydantic model json schema for each tool call you will
make: {'title': 'FunctionCall', 'type': 'object', 'properties': {'arguments': {'title': 'Arguments', 'type': 'object'}, 'name': {'title': 'Name', 'type':
'string'}}, 'required': ['arguments', 'name']}For each function call return a json object with function name and arguments within <tool_call></tool_call>
XML tags as follows:\n<tool_call>\n{tool_call}\n</tool_call>",
"role": "system"
},
{
"content": "Hi, can you tell me the current stock price of Apple?",
"role": "human"
},
{
"content": "<think>Alright, the user asked for the current stock price of Apple. Looking at the tools available, there's a function called
get_stock_price which exactly fits this request. I'll need to call this function with the parameter company set to 'Apple' to retrieve the information
the user is asking for. This seems like the most straightforward and appropriate action to take.\n</think><tool_call>\n{'name': 'get_stock_price',
'arguments': {'company': 'Apple'}}\n</tool_call>",
"role": "model"
},
{
"content": "<tool_response>\n{'stock_price': '$150.75'}\n</tool_response>",
"role": "tool"
},
{
"content": "The current stock price of Apple is $150.75.",
"role": "model"
},
```


3570 rows

Source: https://huggingface.co/datasets/Jofthomas/hermes-function-calling-thinking-V1

INTRODUCTION

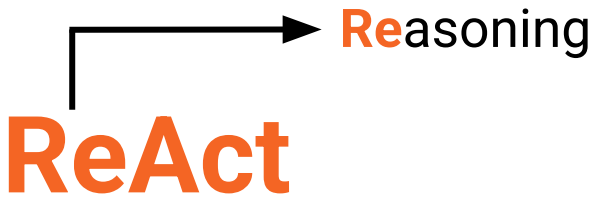
HOW IT WORKS?



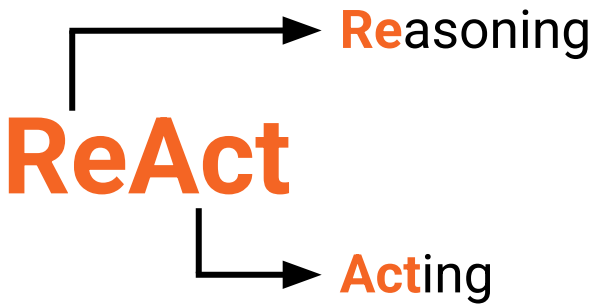
REACT APPROACH

ReAct

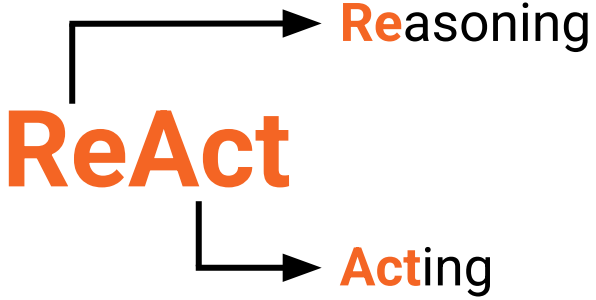
REACT APPROACH



REACT APPROACH



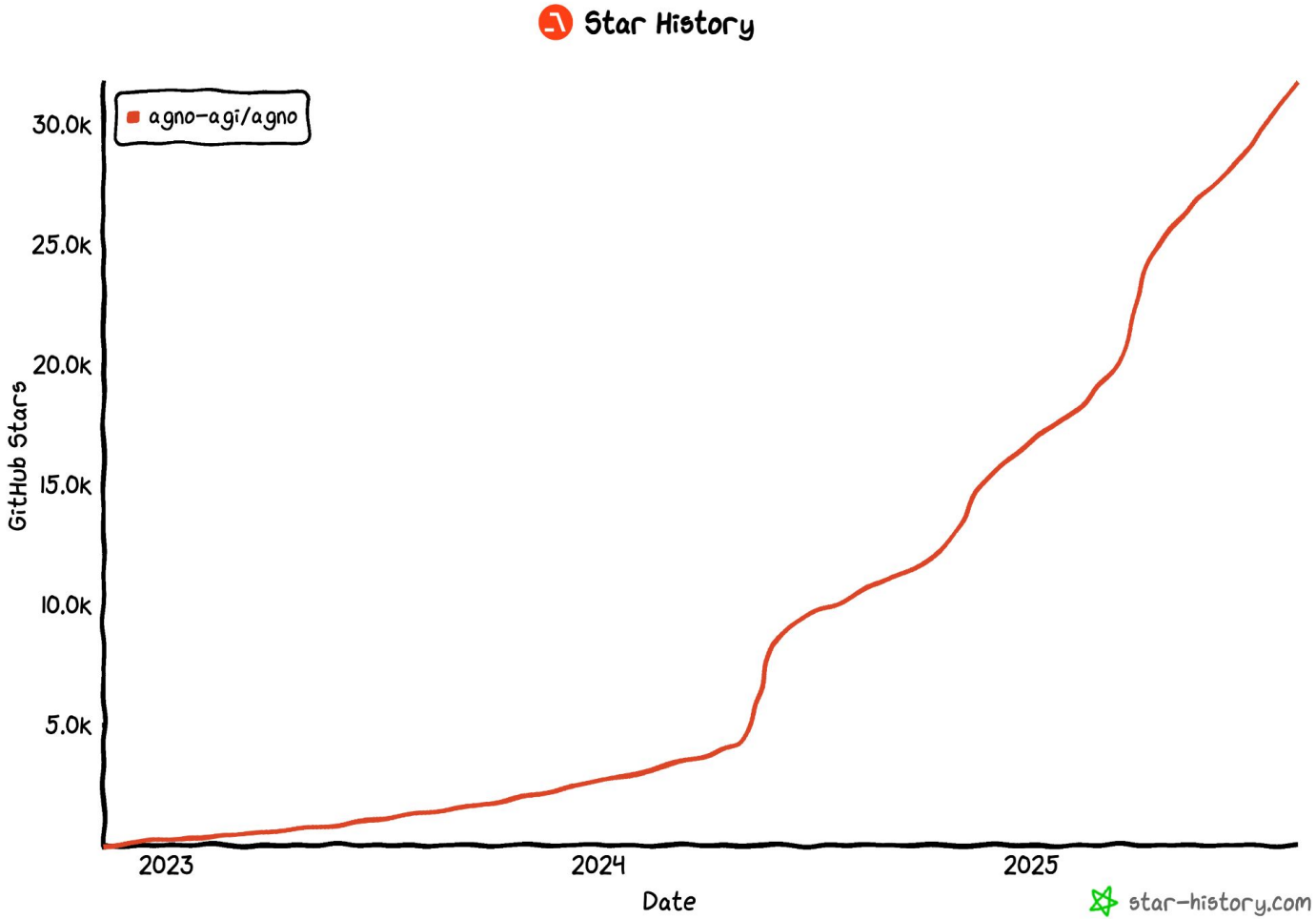
REACT APPROACH



It's a prompting technique that encourages the model to think step-by-step and interleave actions between reasoning steps.

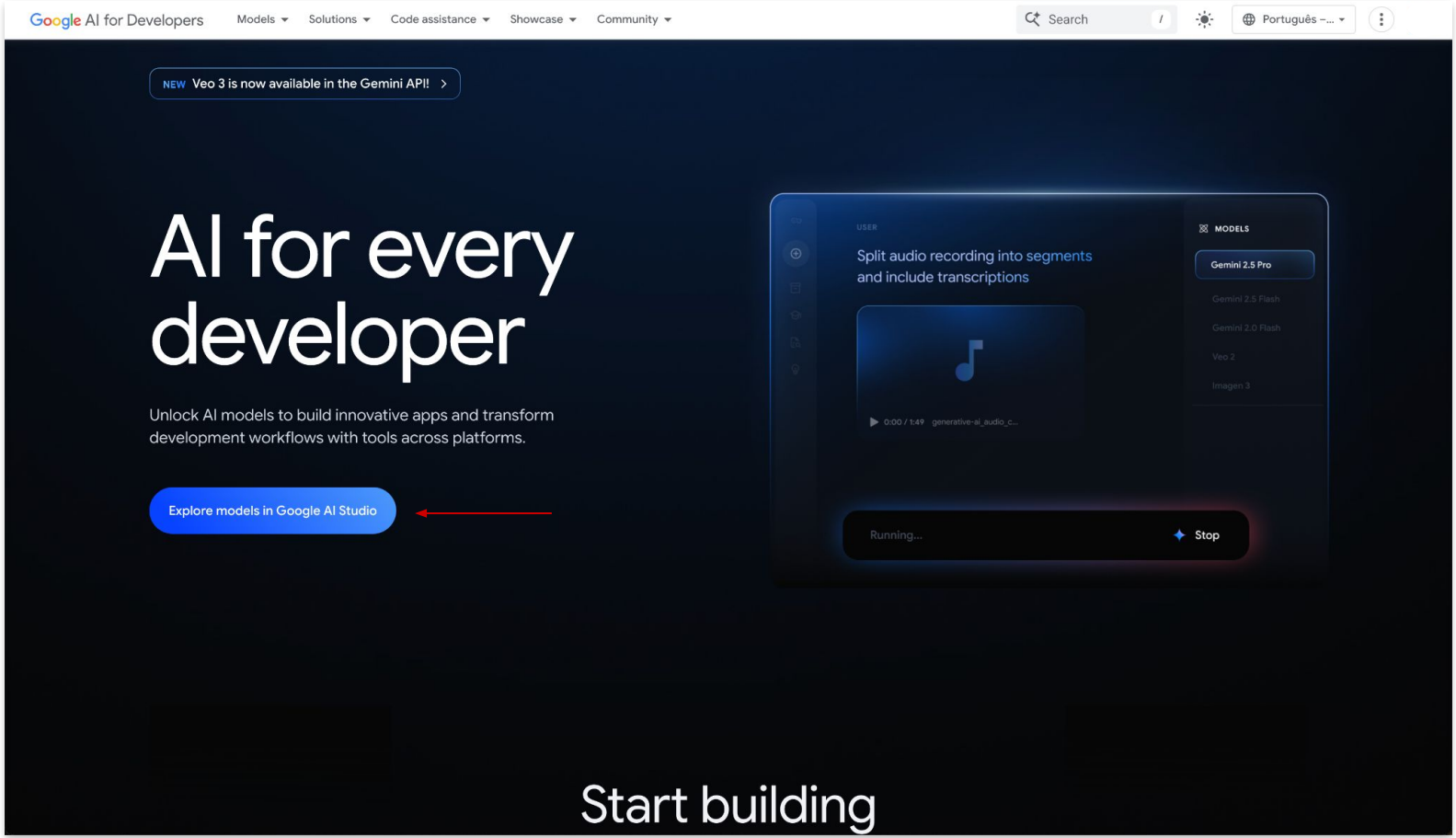
Hugging Face

HANDS-ON #1

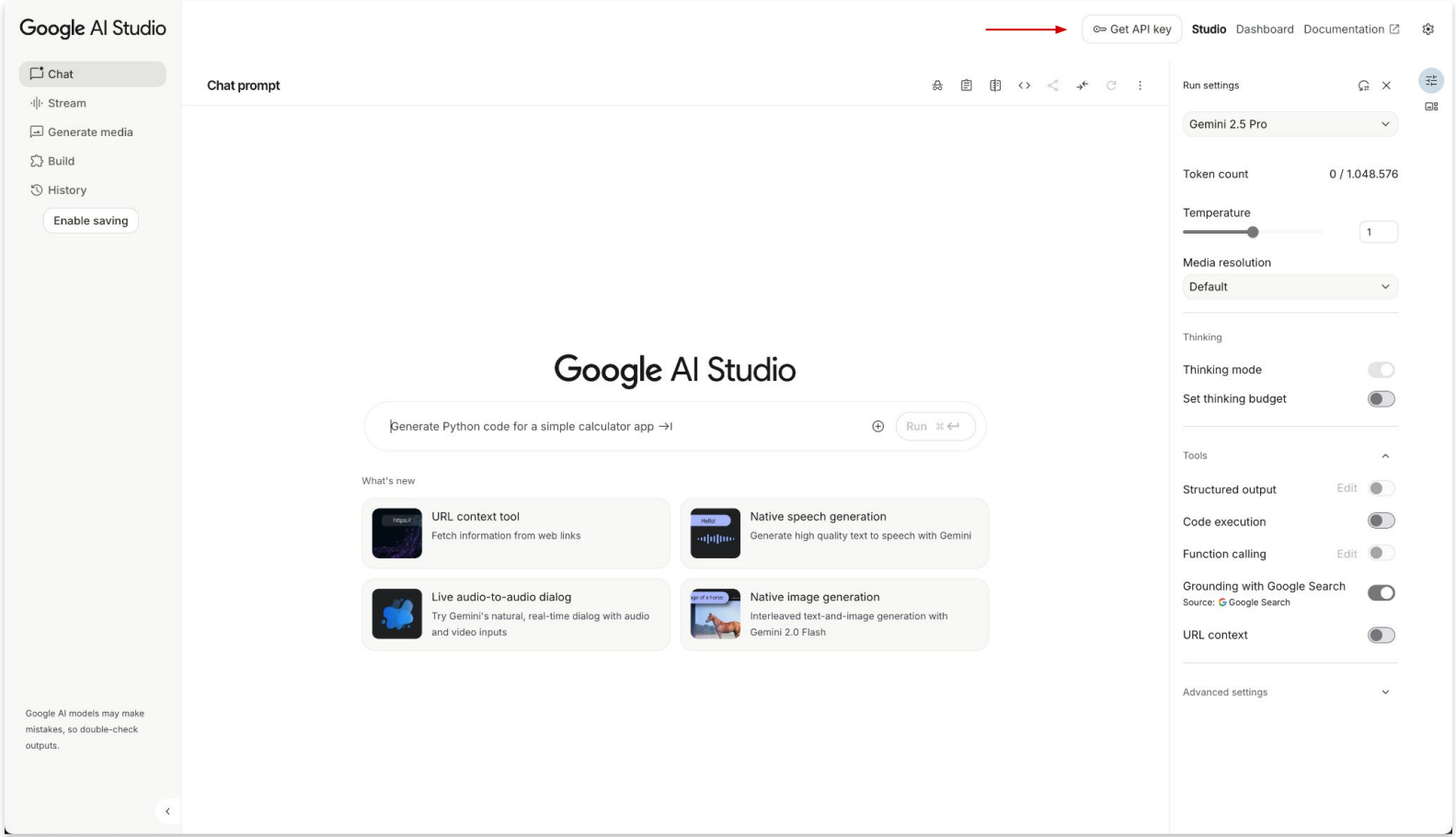

Star History
agno-agi/agno
30.0k
25.0k
20.0k
15.0k
10.0k
5.0k
GitHub Stars
2023
2024
2025
Date
star-history.com


Gemini

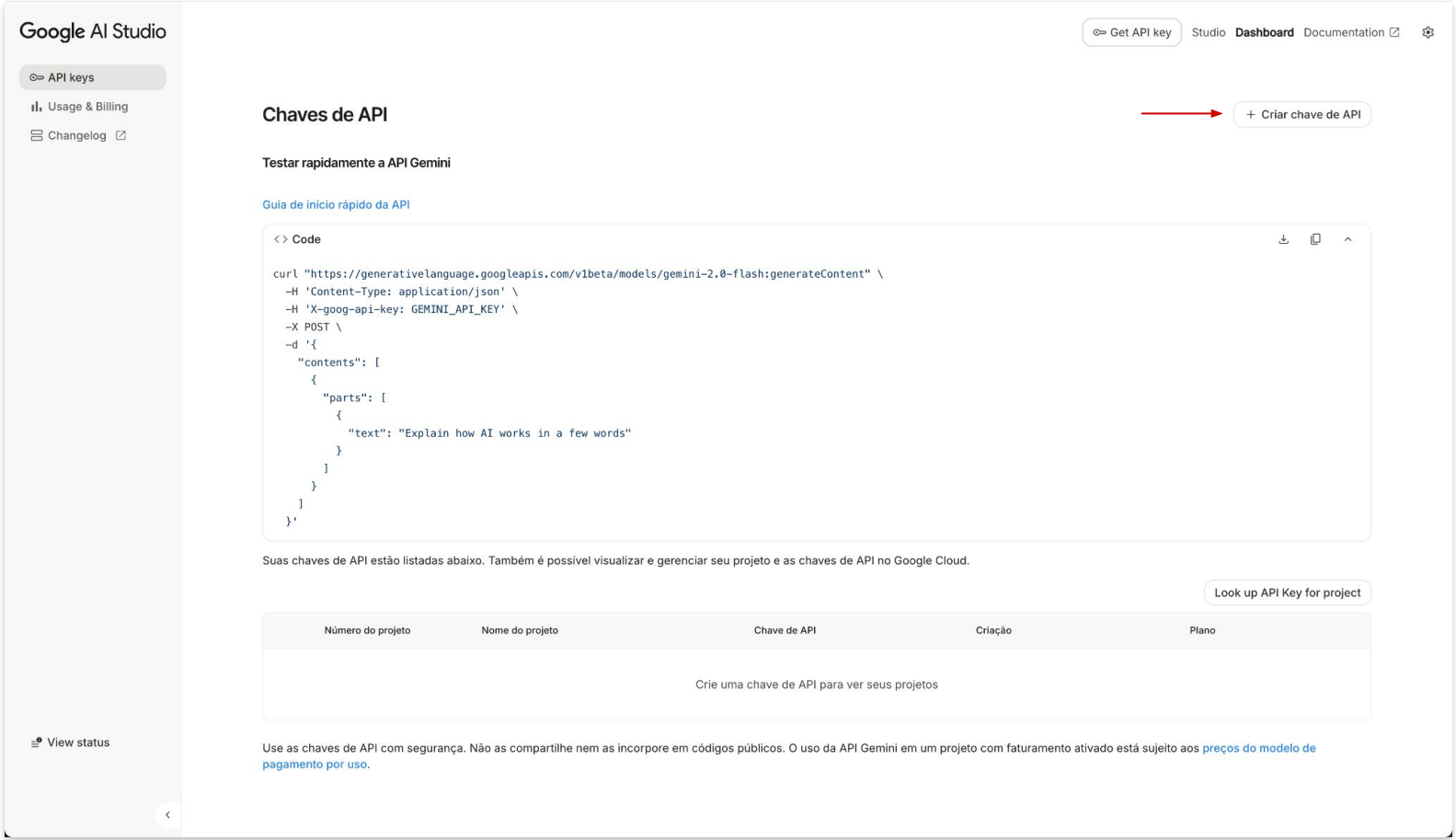
https://ai.google.dev/



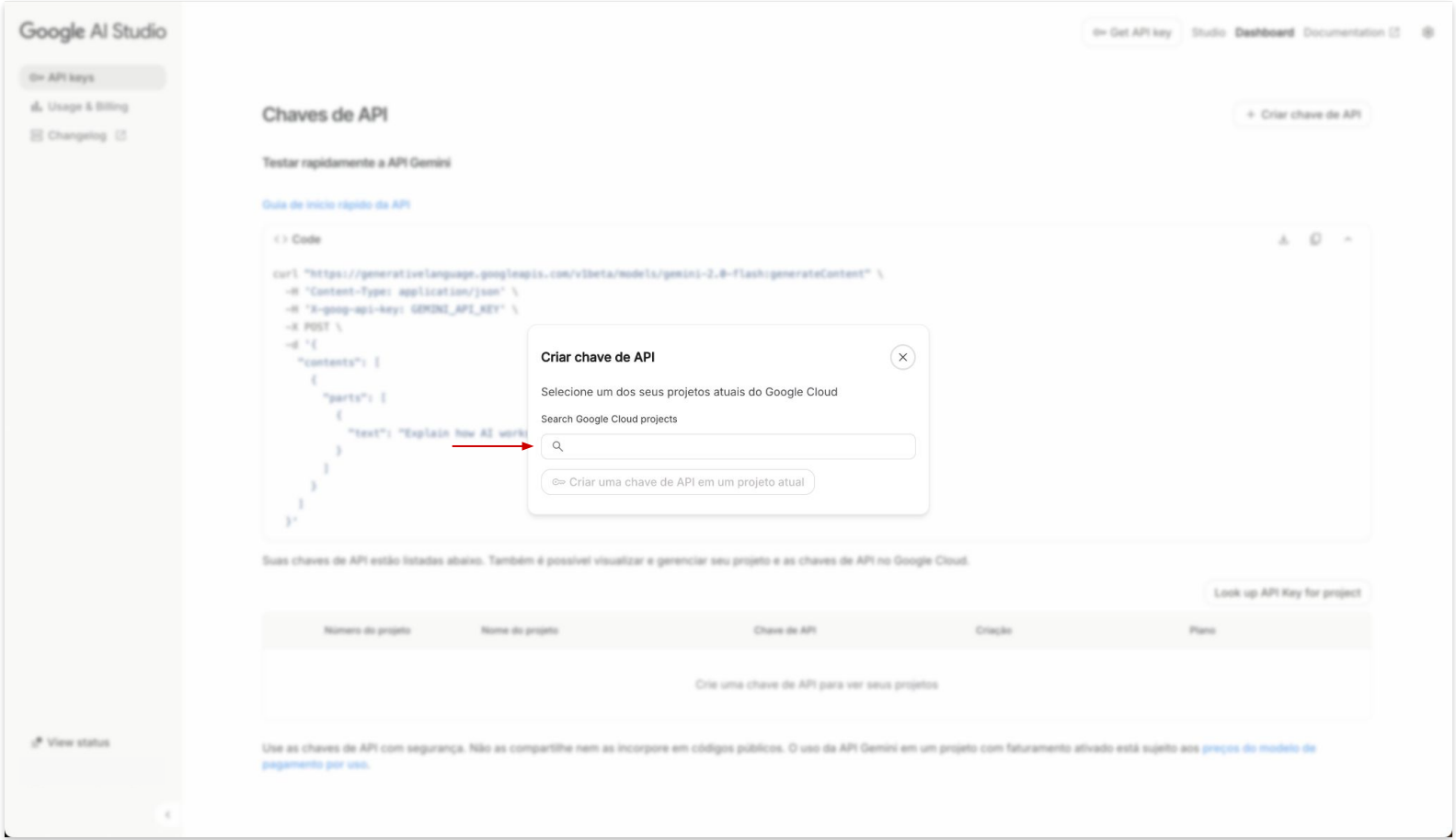
https://ai.google.dev/



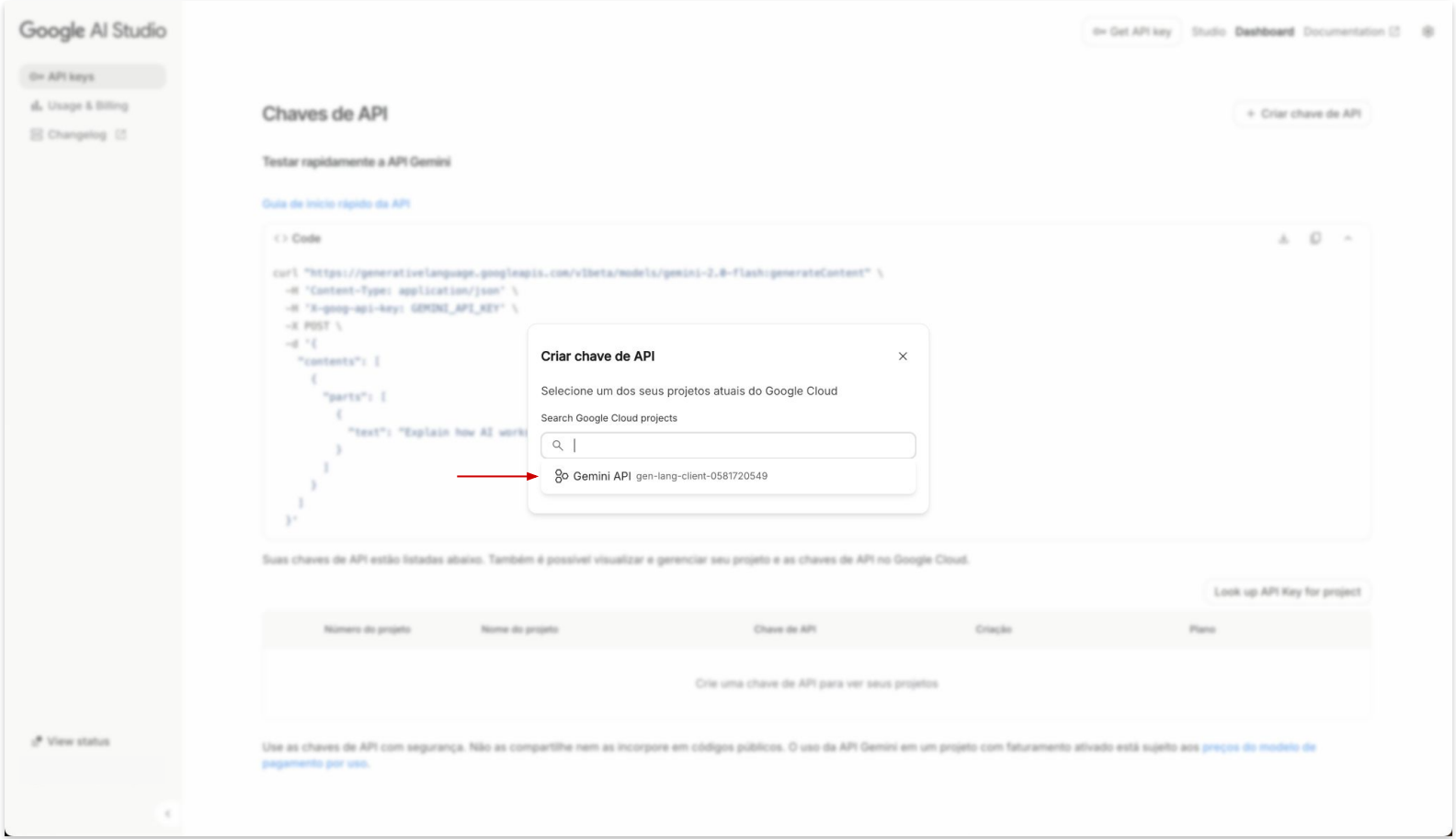
https://ai.google.dev/



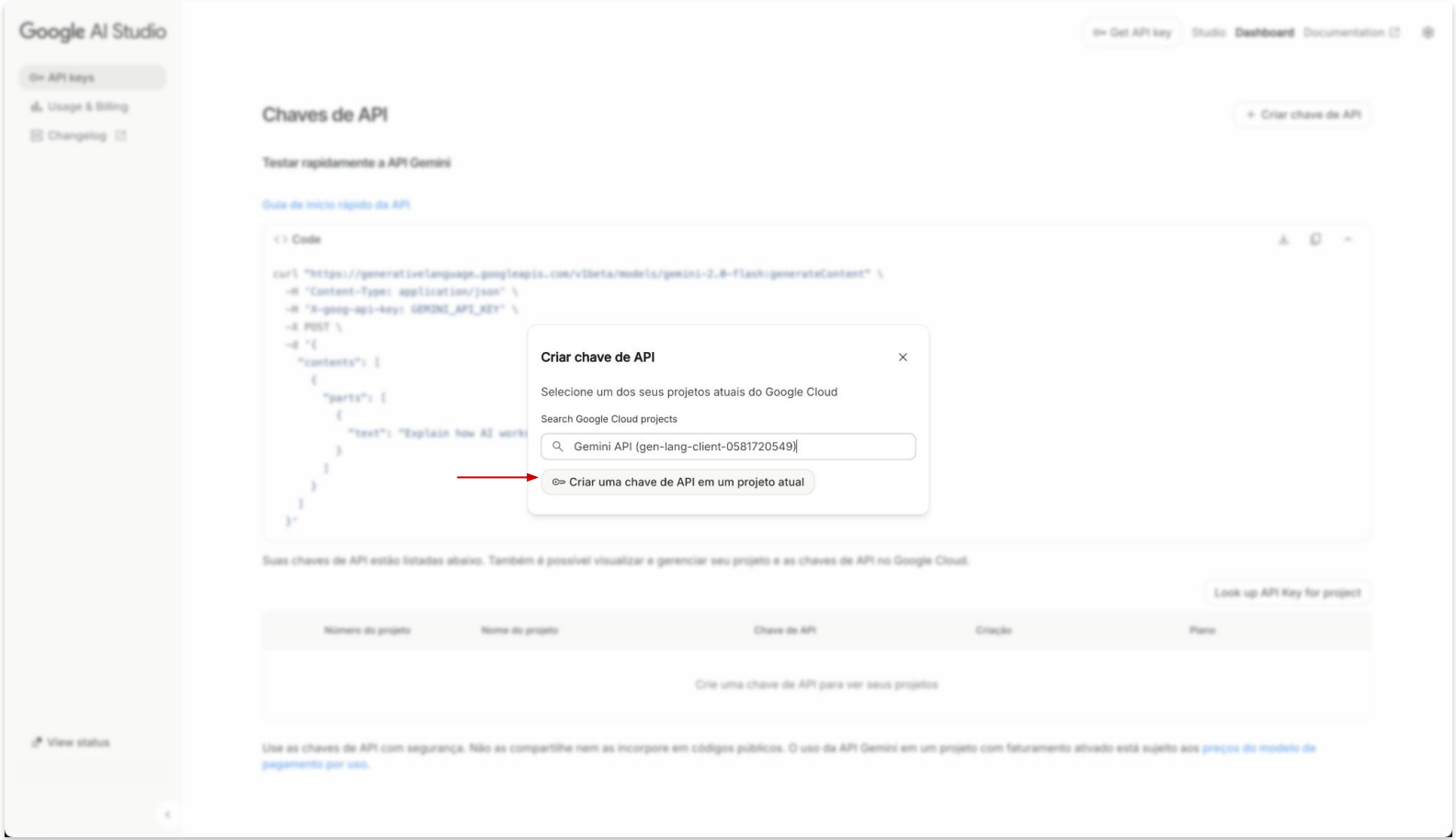
https://ai.google.dev/



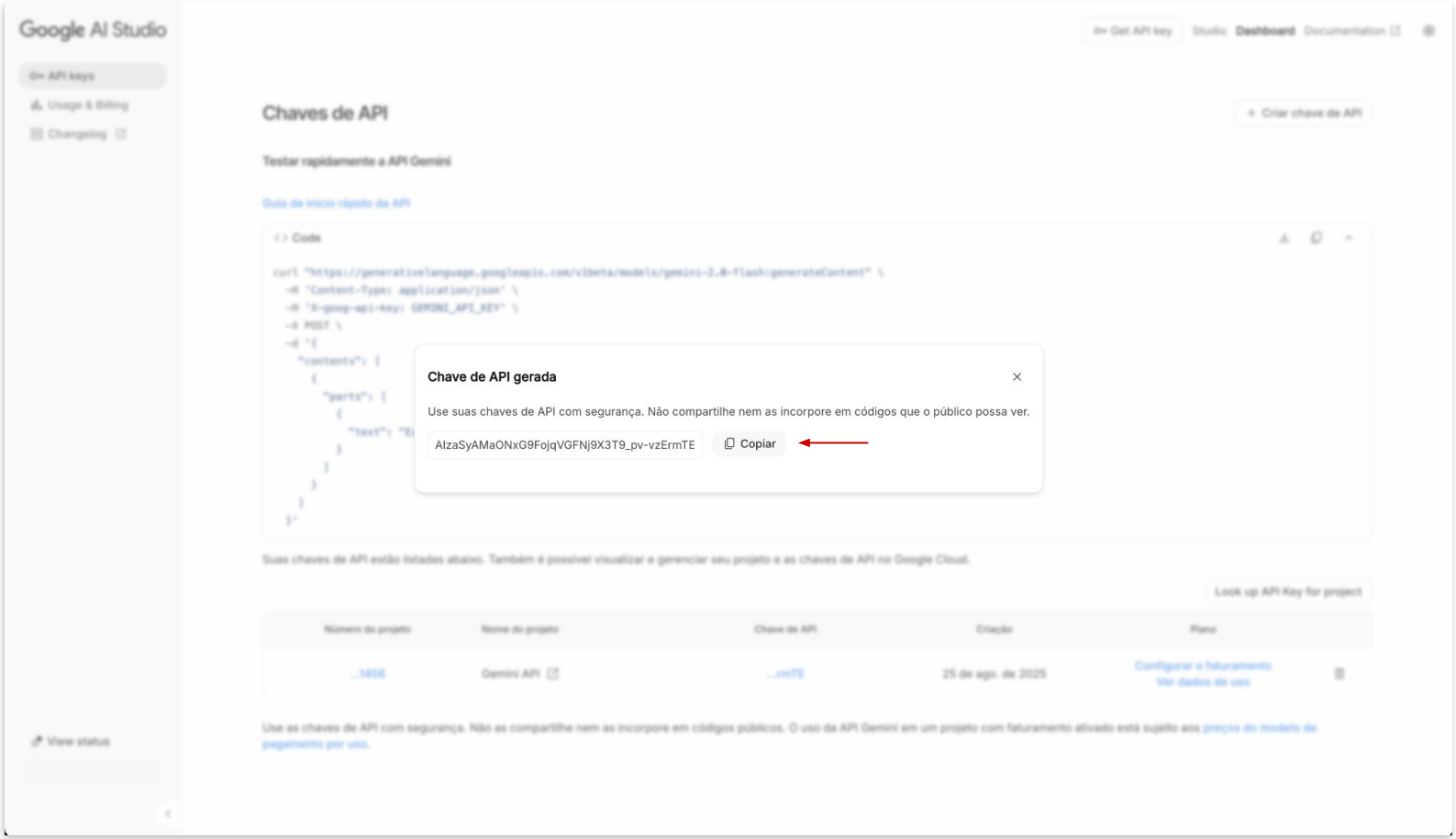
https://ai.google.dev/

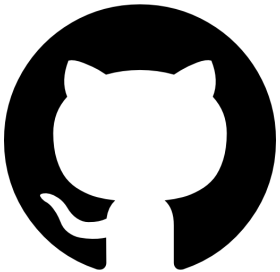


https://ai.google.dev/



https://ai.google.dev/





https://github.com/conect2ai/Agents-Bootcamp

01_HandsOn.ipynb

HANDS-ON
#2


ccee
a ccee
comunicação
dados e análises
preços
mercado
documentos
ccee academy
1º Fórum
ANÁLISE SETORIAL
ccee
Inscrições abertas
Participe
Veja ma
Preço d
Hora vigente 11:0
home
bandeira tarifária
O regime de bandeiras tarifárias é um mecanismo que repassa imediatamente ao consumidor eventuais aumentos nos custos da geração de energia elétrica. Esses recursos são administrados pela CCEE, por determinação da Agência Nacional de Energia Elétrica – Aneel.
Existem três tipos de bandeiras tarifárias, que adotam uma simbologia parecida com um semáforo: elas são divididas nas cores verde, amarela e vermelha.
A verde indica que as condições para a geração de energia são favoráveis e que, portanto, não é preciso fazer cobranças adicionais. A amarela sinaliza que as circunstâncias não são tão cômodas e que é necessário cobrar um acréscimo na energia consumida. E a vermelha aponta o quadro mais desfavorável, com um custo de produção muito elevado, sendo inevitável recolher uma tarifa adicional.
Os consumidores podem descobrir qual é a bandeira tarifária em vigor procurando a informação no boleto mensal de cobrança.

info **bandeiras**tarifárias

ccee

Tabela 11 – Número de horas de cada semana e patamar de carga para o mês de agosto de 2025

Patamar	Nº de Horas por Patamar nas Semanas Operativas					
	Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6
Pesado	5	25	25	25	25	0
Médio	11	65	65	65	65	10
Leve	8	78	78	78	78	38
Total	24	168	168	168	168	48

Os valores de PLD para o cálculo do PLD gatilho, considerando a média mensal são obtidos da simulação do DECOMP da primeira semana operativa de agosto de 2025 de cálculo do PLD, o mesmo que estabelece a função de custo futuro para o modelo DESSEM da primeira semana operativa do mês, e uma expectativa de PLD para as próximas semanas[8]. Estes valores de expectativa de PLD para cada semana são apresentados na Tabela 12.

Tabela 12 – Resultado da Função de Custo Futuro do DECOMP para a primeira semana de agosto e expectativa para as demais semanas

Subm.	Patamar	Expectativa de PLD do modelo DECOMP(R$/MWh)						PLDmédio
		Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6	
Sudeste	Pesado	315,19	313,65	317,16	320,43	321,54	322,94	310,30
	Médio	308,68	308,68	309,18	312,37	314,19	316,24	
	Leve	302,61	303,90	305,44	308,04	310,04	312,17	
Sul	Pesado	315,19	313,65	317,16	320,43	321,54	322,94	310,30
	Médio	308,68	308,68	309,18	312,37	314,19	316,24	
	Leve	302,61	303,90	305,44	308,04	310,04	312,17	
Nord.	Pesado	315,19	313,65	317,16	320,43	321,54	322,94	310,30
	Médio	308,68	308,68	309,18	312,37	314,19	316,24	
	Leve	302,61	303,90	305,44	308,04	310,04	312,17	
Norte	Pesado	315,19	313,65	317,16	320,43	321,54	322,94	310,30
	Médio	308,68	308,68	309,18	312,37	314,19	316,24	
	Leve	302,61	303,90	305,44	308,04	310,04	312,17	

Os valores de carga prevista para cada semana[9], utilizados no cálculo da carga média mensal de cada submercado para ponderar o PLD médio mensal de cada submercado, são apresentados na Tabela 13.

Tabela 13 – Expectativa de carga para cada semana e patamar e a média mensal de agosto de 2025 por submercado

Subm.	Patamar	CARGA (MWmed)						Carga Média
		Sem 1	Sem 2	Sem 3	Sem 4	Sem 5	Sem 6	

Considerando os valores para agosto de 2025, podemos observar as faixas para acionamento das bandeiras tarifárias, após aplicar o cálculo do PLD_{liminf_pat} e PLD_{limsup_pat}. A Tabela 16 apresenta os limites de PLD para o acionamento das bandeiras tarifárias:

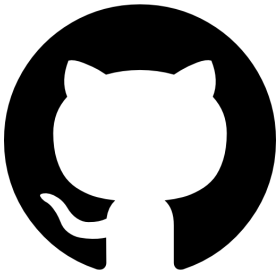
Cor da Bandeira	Gatilho
Verde	$R\$\,58{,}6/MWh <= PLD <= R\$\,74{,}27/MWh$
Amarela	$R\$\,74{,}28/MWh < PLD <= R\$\,186{,}46/MWh$
Vermelha 1	$R\$\,186{,}47/MWh < PLD <= R\$\,256{,}9/MWh$
Vermelha 2	$R\$\,256{,}91/MWh < PLD <= R\$\,751{,}73/MWh$

mostrado na Tabela 17.

Tabela 17 – Cor da Bandeira Tarifária de agosto de 2025

Cor da Bandeira Tarifária	
Agosto de 2025	Vermelha 2

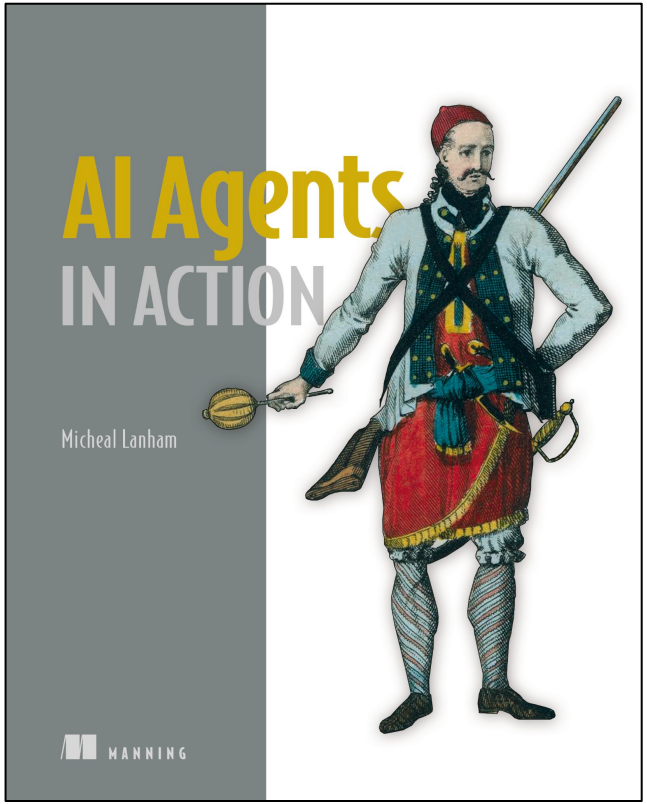
Nota 1: Cabe destacar que o cálculo do PLD_{liminf_pat} e PLD_{limsup_pat} consideram os valores apresentados na Figura 5 da Nota Técnica nº 021/2021-SRG-SGT-SRM/ANEEL e o valor de GSF_{band} obtido pela proporção de GH_{band}/GF_{band} é considerado com arredondamento em duas casas decimais, seguindo solicitação da ANEEL.

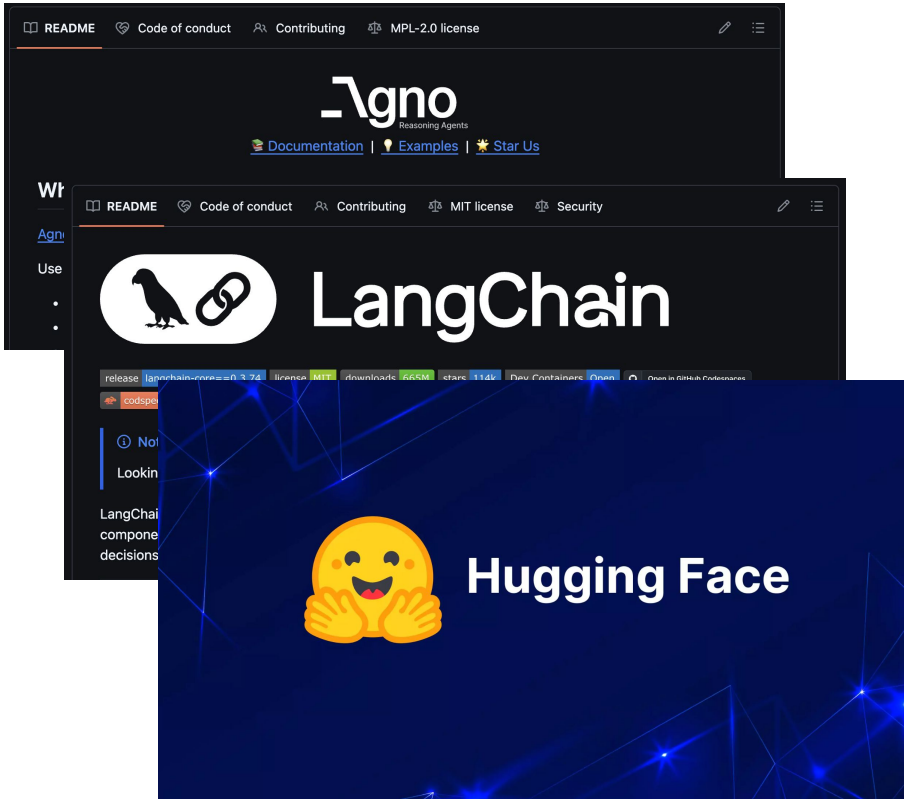


https://github.com/conect2ai/Agents-Bootcamp

02_HandsOn.ipynb

REFERENCES


AI Agents
IN ACTION
Micheal Lanham
MANNING


README
Code of conduct
Contributing
MPL-2.0 license
Documentation
Examples
Star Us
README
Code of conduct
Contributing
MIT license
Security
LangChain
Hugging Face

THANKS!

matheus.diniz.122@ufrn.edu.br

@conect2ai / @dinizmaths

C2AI

CAECOMP

IEEE COMPUTER SOCIETY