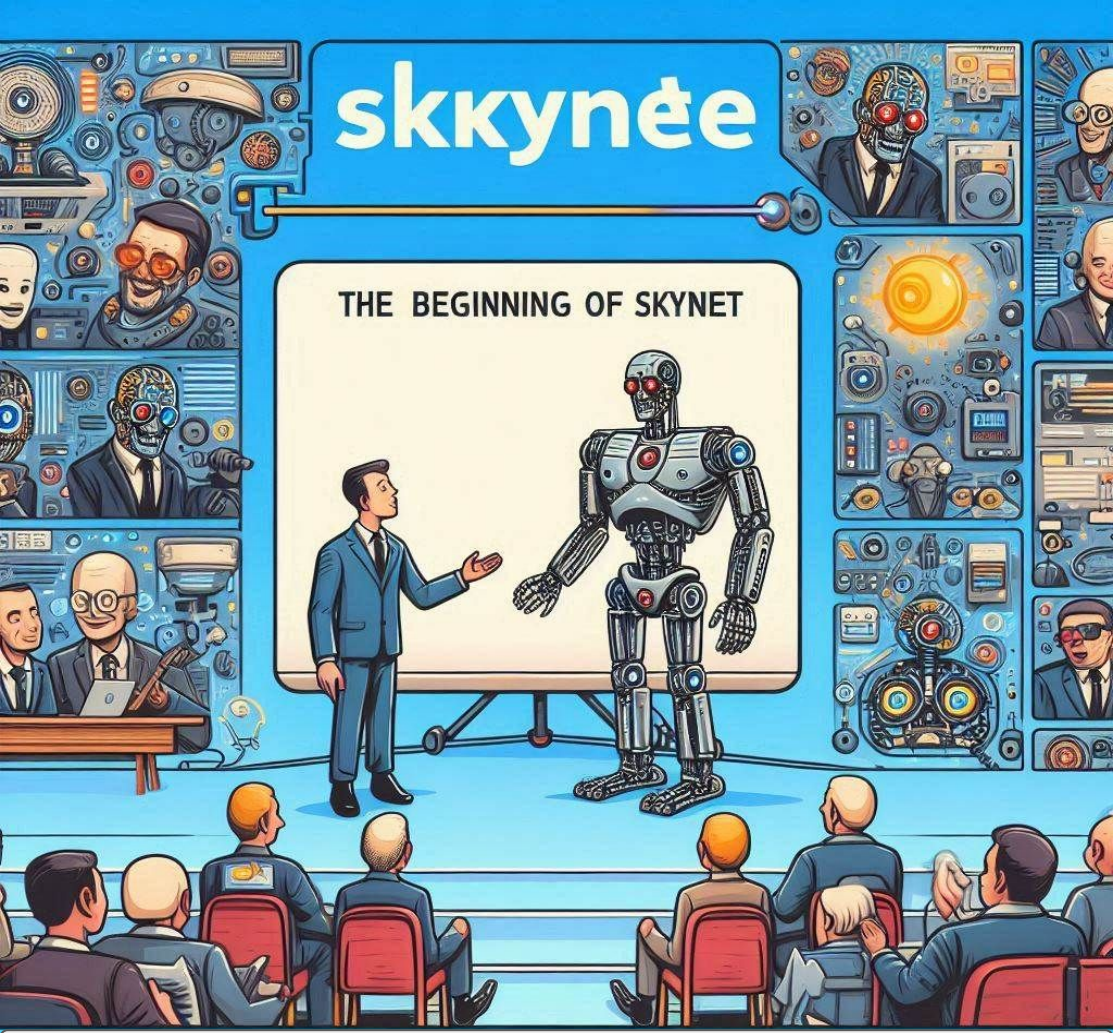


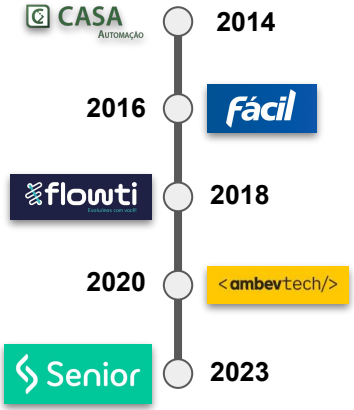
Prompt: Capa para slides de apresentação sobre o início da skynet usando ia generativa

Dando início a Skynet com LLamaIndex

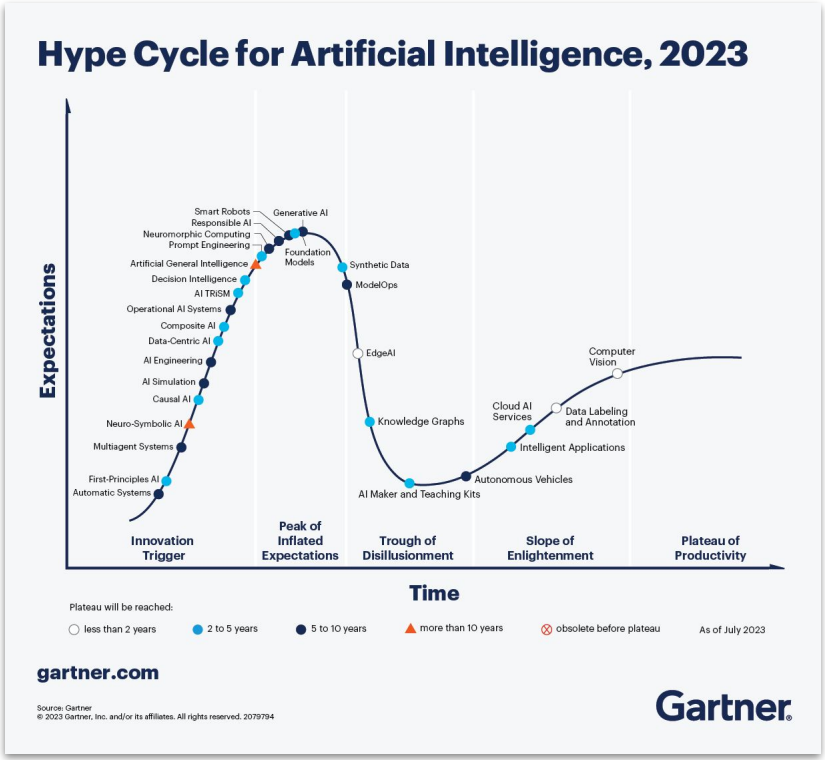
José Henrique Luckmann

Apresentação

- Acadêmico de Ciência da Computação
- Trabalho com programação desde 2014
- Me especializei na área de dados em 2018
- Atualmente trabalho com pesquisa aplicada



Por que IA Generativa se tornou importante?



Gizmodo

IA generativa nos negócios: pesquisa aponta que empresas brasileiras se preparam

Levantamento com mais de 100 empresas de grande porte do Brasil mostrou a relação delas com IA generativa.

23 horas atrás

Inforchannel

Alteryx AiDIN ganha novos recursos de IA generativa com AI Studio

O novo Alteryx AI Studio foi projetado para dar às organizações o poder de selecionar um LLM de sua escolha em um lista de opções...

1 dia atrás

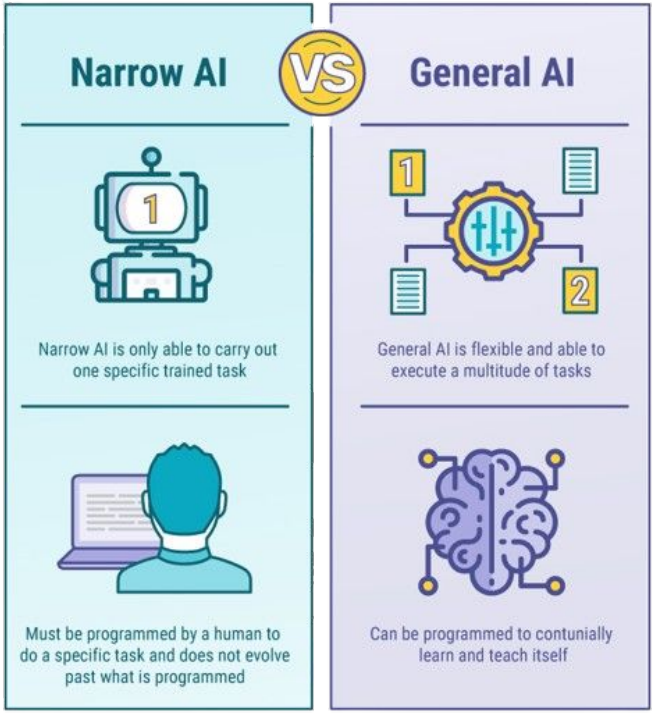
Forbes Brasil

Meta disponibiliza IA generativa para anunciantes

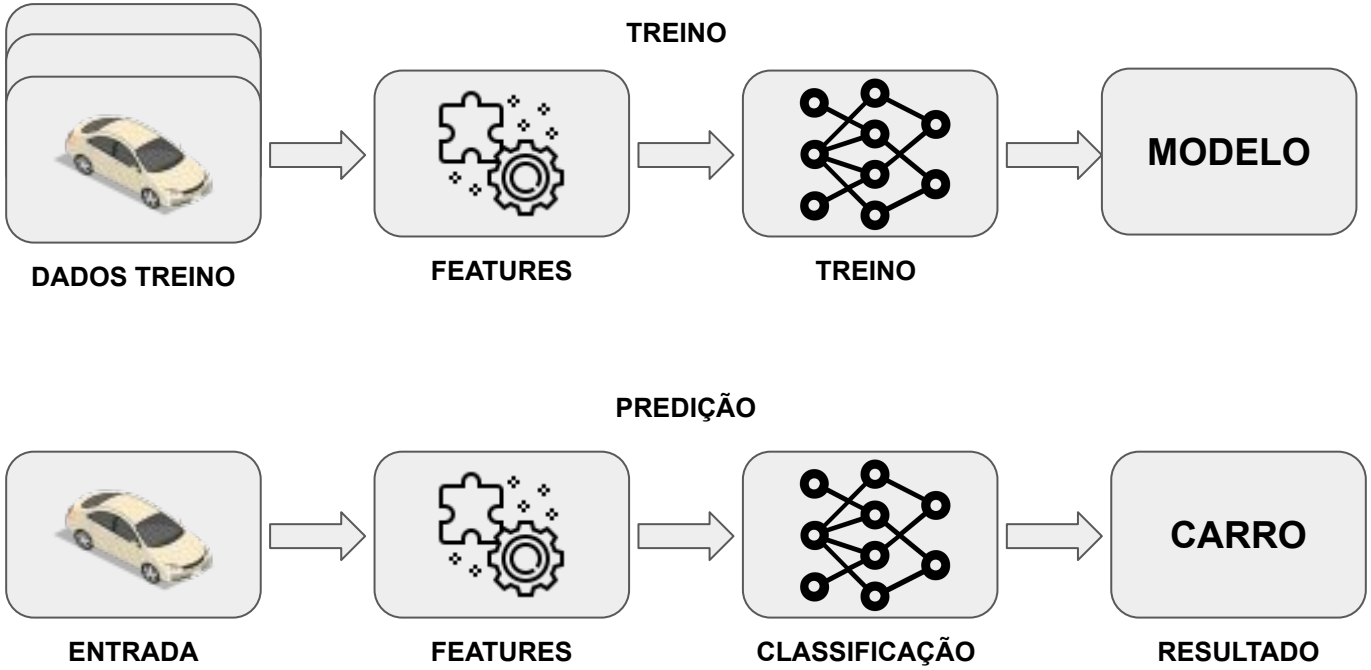
"Nova ferramenta explora vastos conjuntos de dados antigos para gerar conteúdo novo, como arte e código de software.

2 dias atrás

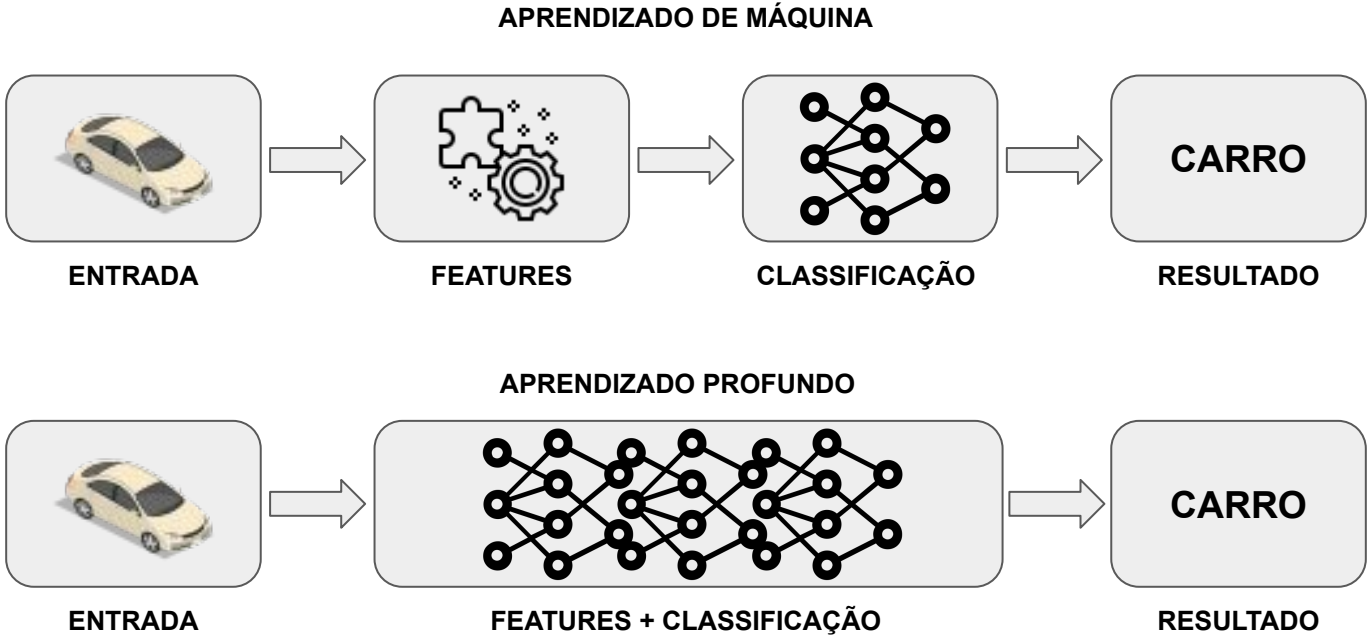
O que é IA Generativa



Aprendizado de Máquina



Aprendizado Profundo



Artigos relevantes

A fast learning algorithm for deep belief nets *

Geoffrey E. Hinton and Simon Osindero
Department of Computer Science University of Toronto
10 Kings College Road
Toronto, Canada M5S 3G4
{hinton, osindero}@cs.toronto.edu

Yee-Whye Teh
Department of Computer Science
National University of Singapore
3 Science Drive 3, Singapore, 117543
tehyw@comp.nus.edu.sg

Abstract

We show how to use "complementary priors" to eliminate the explaining away effects that make inference difficult in densely-connected belief nets that have many hidden layers. Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory. The fast, greedy algorithm is used to initialize a slower learning procedure that fine-tunes the weights using a contrastive version of the wake-sleep algorithm. After fine-tuning, a network with three hidden layers forms a very good generative model of the joint distribution of handwritten digit images and their labels. This generative model gives better digit classification than the best discriminative learning algorithms. The low-dimensional manifolds on which the digits lie are modelled by long ravines in the free-energy landscape of the top-level associative memory and it is easy to explore these ravines by using the directed connections to display what the associative memory has in mind.

1 Introduction

Learning is difficult in densely-connected, directed belief nets that have many hidden layers because it is difficult to infer the conditional distribution of the hidden activities when given a data vector. Variational methods use simple approximations to the true conditional distribution, but the approximations may be poor, especially at the deepest hidden layer where the prior assumes independence. Also, variational learning still requires all of the parameters to be learned together and makes the learning time scale poorly as the number of parameters increases.

We describe a model in which the top two hidden layers form an *undirected* associative memory (see figure 1) and the remaining hidden layers form a directed acyclic graph that converts the representations in the associative memory into observable variables such as the pixels of an image. This hybrid model has some attractive features:

1. There is a fast, greedy learning algorithm that can find a fairly good set of parameters quickly, even in deep networks with millions of parameters and many hidden layers.
2. The learning algorithm is unsupervised but can be applied to labeled data by learning a model that generates both the label and the data.
3. There is a fine-tuning algorithm that learns an excellent generative model which outperforms discriminative methods on the MNIST database of hand-written digits.
4. The generative model makes it easy to interpret the distributed representations in the deep hidden layers.
5. The inference required for forming a percept is both fast and accurate.
6. The learning algorithm is local: adjustments to a synapse strength depend only on the states of the presynaptic and post-synaptic neuron.
7. The communication is simple: neurons only need to communicate their stochastic binary states.

Section 2 introduces the idea of a "complementary" prior which exactly cancels the "explaining away" phenomenon that makes inference difficult in directed models. An example of a directed belief network with complementary priors is presented. Section 3 shows the equivalence between restricted Boltzmann machines and infinite directed networks with tied weights.

Section 4 introduces a fast, greedy learning algorithm for constructing multi-layer directed networks one layer at a time. Using a variational bound it shows that as each new layer is added, the overall generative model improves. The greedy algorithm bears some resemblance to boosting in its repeated use of the same "weak" learner, but instead of reweighting each data-vector to ensure that the next step learns something new, it re-represents it. The "weak" learner that

* To appear in Neural Computation 2006

A fast learning algorithm for deep belief nets∗
Geoffrey E. Hinton and Simon Osinderof

DOI:10.1145/3422622

Generative Adversarial Networks

By Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio

Abstract

Generative adversarial networks are a kind of artificial intelligence algorithm designed to solve the *generative modeling* problem. The goal of a generative model is to study a collection of training examples and learn the probability distribution that generated them. Generative Adversarial Networks (GANs) are then able to generate more examples from the estimated probability distribution. Generative models based on deep learning are common, but GANs are among the most successful generative models (especially in terms of their ability to generate realistic high-resolution images). GANs have been successfully applied to a wide variety of tasks (mostly in research settings) but continue to present unique challenges and research opportunities because they are based on game theory while most other approaches to generative modeling are based on optimization.

1. INTRODUCTION

Most current approaches to developing artificial intelligence are based primarily on machine learning. The most widely used and successful form of machine learning to date is supervised learning. Supervised learning algorithms are given a dataset of pairs of example inputs and example outputs. They learn to associate each input with each output and thus learning a mapping from input to output examples. The input examples are typically complicated data objects like images, natural language sentences, or audio waveforms, while the output examples are often relatively simple. The most common kind of supervised learning is classification, where the output is just an integer code identifying a specific category (a photo might be recognized as coming from category 0 containing cats, or category 1 containing dogs, etc.).

Supervised learning is often able to achieve greater than human accuracy after the training process is complete, and thus has been integrated into many products and services. Unfortunately, the learning process itself still falls far short of human abilities. Supervised learning by definition relies on a human supervisor to provide an output example for each input example. Worse, existing approaches to supervised learning often require millions of training examples to exceed human performance, when a human might be able to learn to perform the task acceptably from a very small number of examples.

In order to reduce both the amount of human supervision required for learning and the number of examples required for learning, many researchers today study *unsupervised learning*, often using *generative models*. In this overview paper, we describe one particular approach to unsupervised learning via generative modeling called *generative adversarial networks*. We briefly review applications of GANs and identify core research problems related to convergence in games necessary to make GANs a reliable technology.

2. GENERATIVE MODELING

The goal of supervised learning is relatively straightforward to specify, and all supervised learning algorithms have essentially the same goal: learn to accurately associate new input examples with the correct outputs. For instance, an object recognition algorithm may associate a photo of a dog with some kind of DOG category identifier.

Unsupervised learning is a less clearly defined branch of machine learning, with many different unsupervised learning algorithms pursuing many different goals. Broadly speaking, the goal of unsupervised learning is to learn something useful by examining a dataset containing unlabeled input examples. Clustering and dimensionality reduction are common examples of unsupervised learning.

Another approach to unsupervised learning is generative modeling. In generative modeling, training examples x are drawn from an unknown distribution $p_{data}(x)$. The goal of a generative modeling algorithm is to learn a $p_{model}(x)$ that approximates $p_{data}(x)$ as closely as possible.

A straightforward way to learn an approximation of p_{data} is to explicitly write a function $p_{model}(x;\theta)$ controlled by parameters θ and search for the value of the parameters that makes p_{data} and p_{model} as similar as possible. In particular, the most popular approach to generative modeling is probably *maximum likelihood estimation*, consisting of minimizing the Kullback-Leibler divergence between p_{data} and p_{model}. The common approach of estimating the mean parameter of a Gaussian distribution by taking the mean of a set of observations is one example of maximum likelihood estimation. This approach based on explicit density functions is illustrated in Figure 1.

Explicit density modeling has worked well for traditional statistics, using simple functional forms of probability distributions, usually applied to small numbers of variables. More recently, with the rise of machine learning in general and deep learning in particular, researchers have become interested in learning models that make use of relatively complicated functional forms. When a deep neural network is used to generate data, the corresponding density function may be computationally intractable. Traditionally, there have been two dominant approaches to confronting this intractability problem: (1) carefully design the model to have a tractable density function (e.g., Frey[4]) and (2) design a learning algorithm based on

The original version of this paper is entitled "Generative Adversarial Networks" and was published in *Advances in Neural Information Processing Systems 27* (NIPS 2014).

Generative Adversarial Networks
Ian Goodfellow

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 efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

† Work performed while at Google Brain.

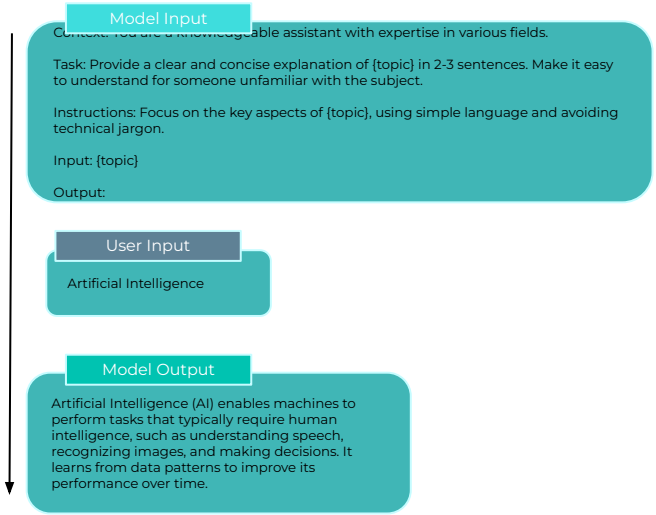
‡ Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

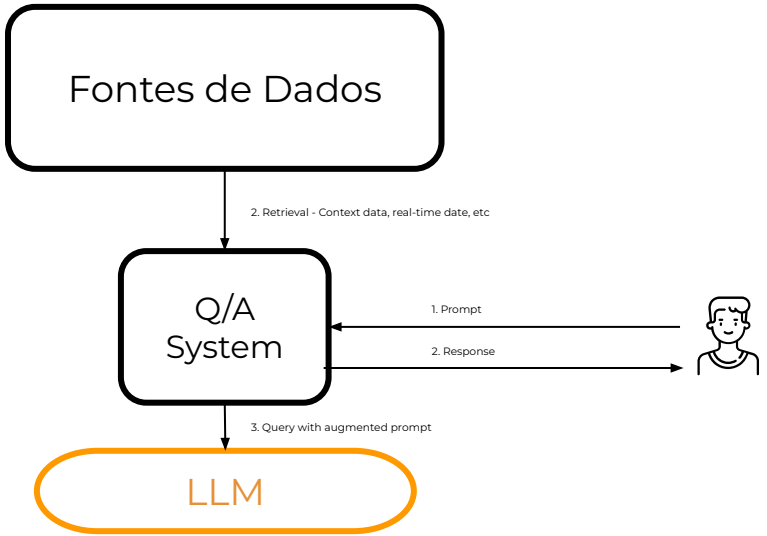
Attention Is All You Need
Ashish Vaswani e Noam Shazeer

Trabalhando com LLMs

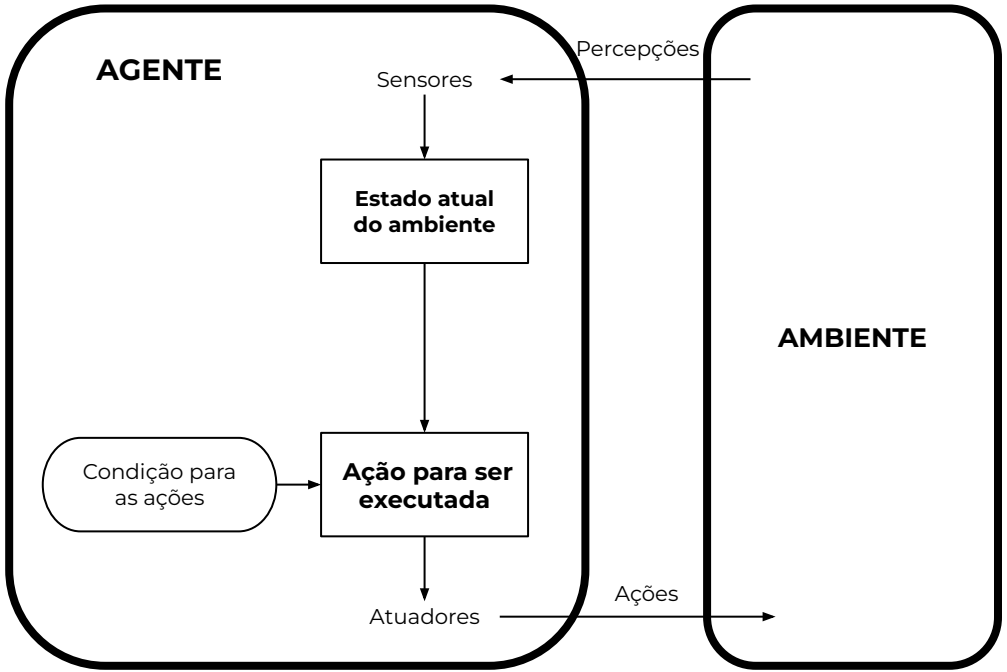
Prompt Engineering



RAG

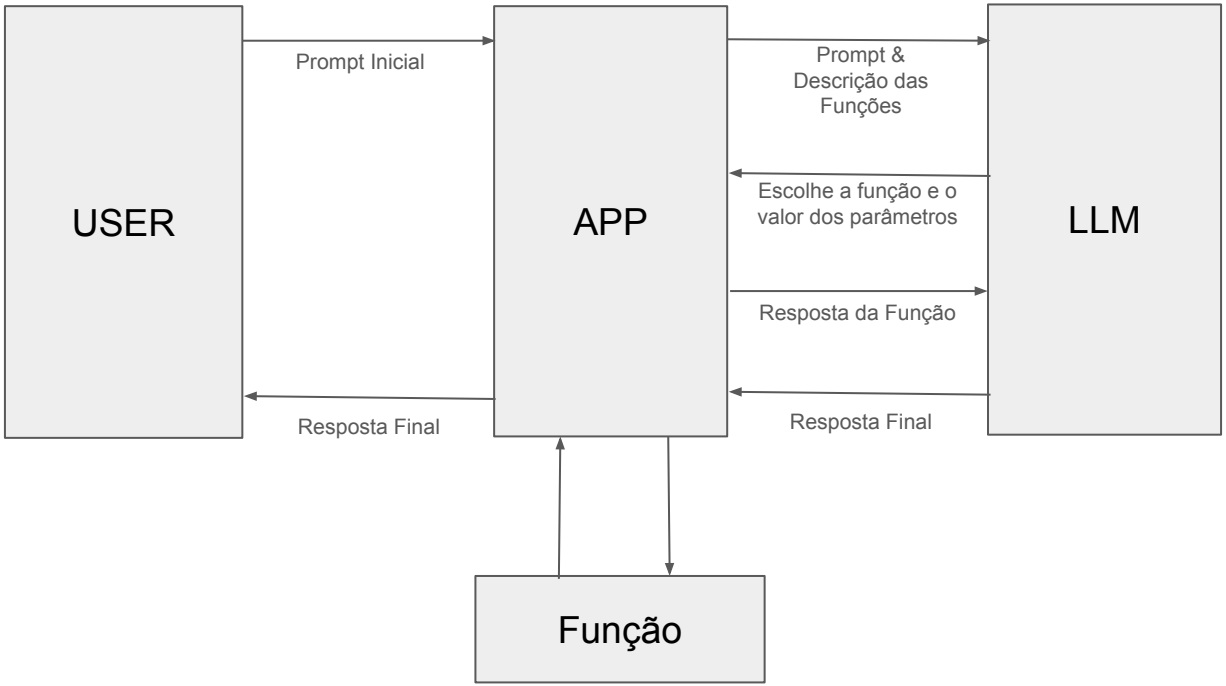


Definição de Agentes



“Em Inteligência Artificial, um agente inteligente é uma entidade autônoma que é capaz de observar um ambiente através de sensores e atuar sobre este através de atuadores.”

Chamada de Funções com LLM



ReAct Agent

Published as a conference paper at ICLR 2023

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

ABSTRACT

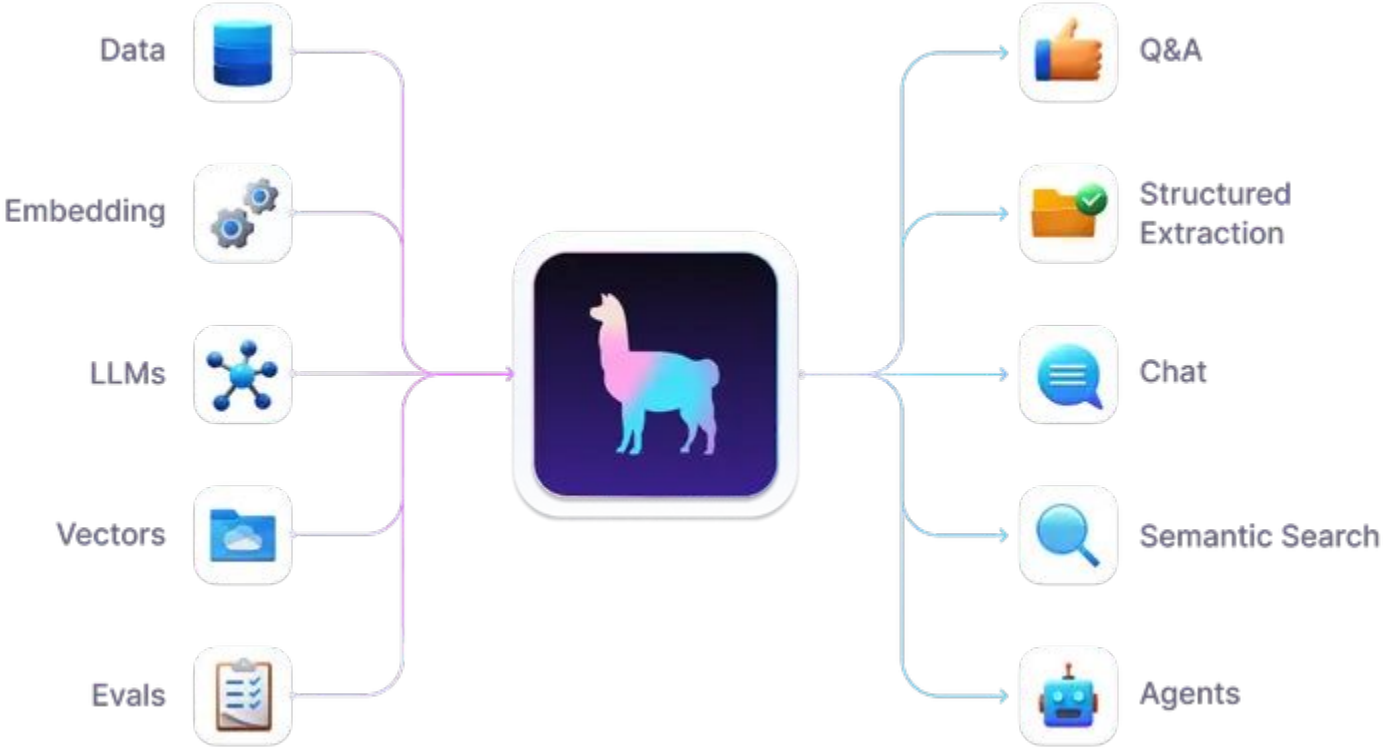
While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. Furthermore, on two interactive decision making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, while being prompted with only one or two in-context examples.

1 INTRODUCTION

A unique feature of human intelligence is the ability to seamlessly combine task-oriented actions with verbal reasoning (or inner speech, Alderson-Day & Fernyhough, 2015), which has been theorized to play an important role in human cognition for enabling self-regulation or strategization (Vygotsky, 1987; Luria, 1965; Fernyhough, 2010) and maintaining a working memory (Baddeley, 1992). Consider the example of cooking up a dish in the kitchen. Between any two specific actions, we may reason in language in order to track progress ("now that everything is cut, I should heat up the pot of water"), to handle exceptions or adjust the plan according to the situation ("I don't have salt, so let me use soy sauce and pepper instead"), and to realize when external information is needed ("how do I prepare dough? Let me search on the Internet"). We may also act (open a cookbook to read the recipe, open the fridge, check ingredients) to support the reasoning and to answer questions ("What dish can I make right now?"). This tight synergy between "acting" and "reasoning" allows humans to learn new tasks quickly and perform robust decision making or reasoning, even under previously

REAC T: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

LLama Index


Data
Embedding
LLMs
Vectors
Evals
Q&A
Structured Extraction
Chat
Semantic Search
Agents

Sql markdown response #16103

Edit | Code

Merged logan-markewich merged 6 commits into `run-llama:main` from `JoseLuckmann:sql-markdown-response` on Sep 20

Conversation 3 | Commits 6 | Checks 10 | Files changed 2 | +43 −2

JoseLuckmann commented on Sep 19 — Contributor

Description

A flag has been implemented to provide functionality where, when this flag is set to True alongside the synthesise_response set to False, instead of returning the data in an array format, it returns the data as a markdown table structure. This is useful for direct responses and aids in processing and understanding the model, as it also provides column names, unlike the previous model's return format.

New Package?

Reviewers
logan-markewich ✓

Assignees
logan-markewich

Labels
lgtm size:M

User: How many r's have in strawberry?

LLM: There are 4 r's in the word "strawberry".

```
def count_letters(word: string) -> dict:
    """Returns a dictionary with the count of each letter in
'word'."""
    return {letter: word.count(letter) for letter in set(word)}

count_letters_tool = FunctionTool.from_defaults(fn=count_letters)

agent = ReActAgent.from_tools(tools=[count_letters_tool], verbose=True)
```

```
User: How many r's have in strawberry?
    > Running step
    Thought: The current language of the user is: English.
    I need to use a tool to help me answer the question.
    Action: count_letters
    Action Input: {'word': 'strawberry'}
    Observation: {'s': 1, 'y': 1, 'e': 1, 't': 1, 'w': 1,
    'a': 1, 'r': 3, 'b': 1}
    > Running step
    Thought: I can answer without using any more tools.
    I'll use the user's language to answer
    Answer: There are 3 'r's in the word "strawberry".
LLM: There are 3 'r's in the word "strawberry".
```

```
target_docs = SimpleDirectoryReader(input_dir="./data/targets/")
target_docs = target_docs.load_data()

target_index = VectorStoreIndex.from_documents(target_docs)
target_engine = target_index.as_query_engine()

target_query_engine = QueryEngineTool(
        query_engine=target_engine,
        metadata=ToolMetadata(
            name="targets",
            description=(
                "Provide information on Skynet's targets. Each document
contains information about a potential target"
                "It receives exactly the question as input and returns
the most relevant answer."
            ),
        ),
    )
```

Você sabe qual o significado da palavra tezgüino?

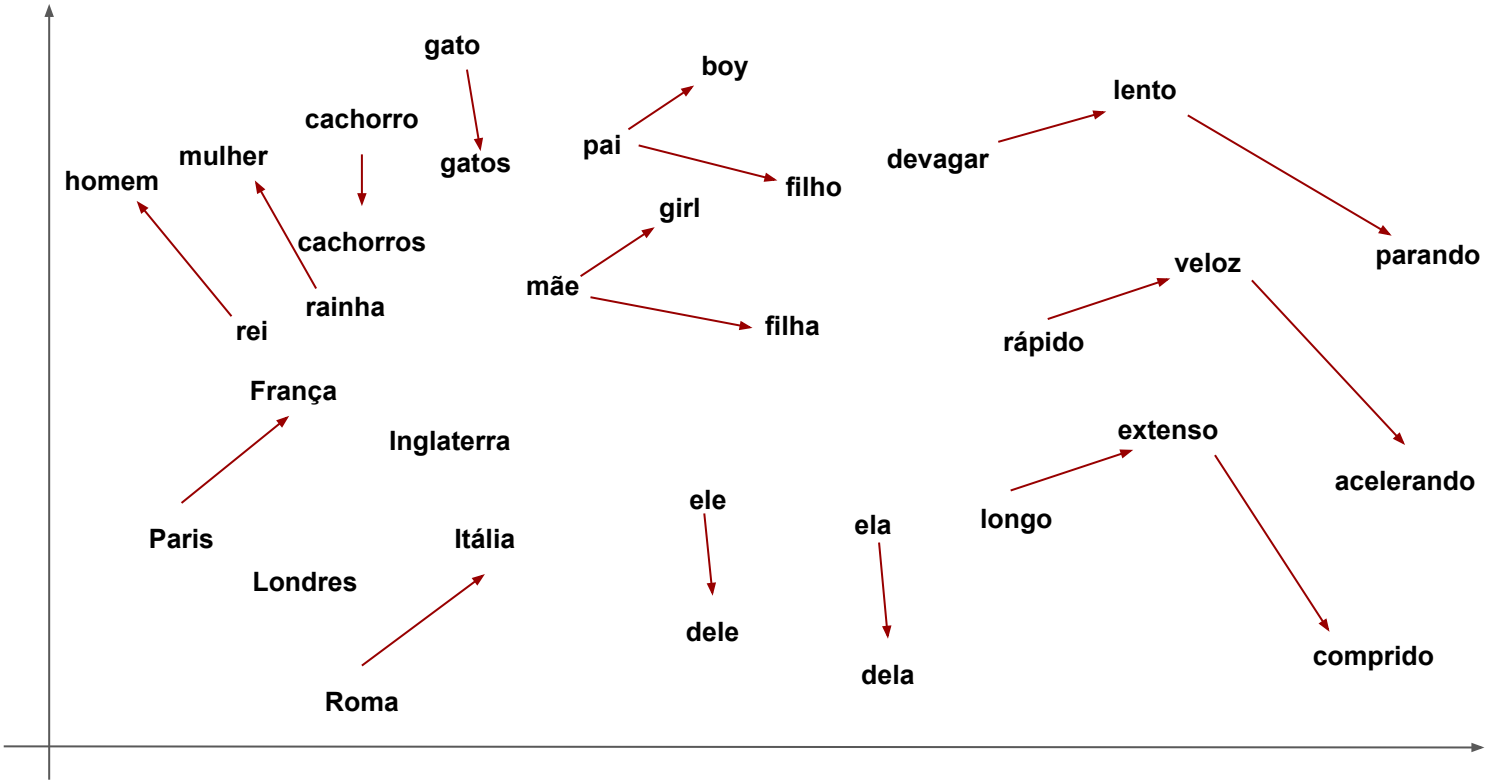
1. Uma garrafa de tezgüino está sobre a mesa.
2. Todo mundo gosta de beber tezgüino.
3. Você pode ficar bêbado com tezgüino.
4. Tezgüino é feito de milho

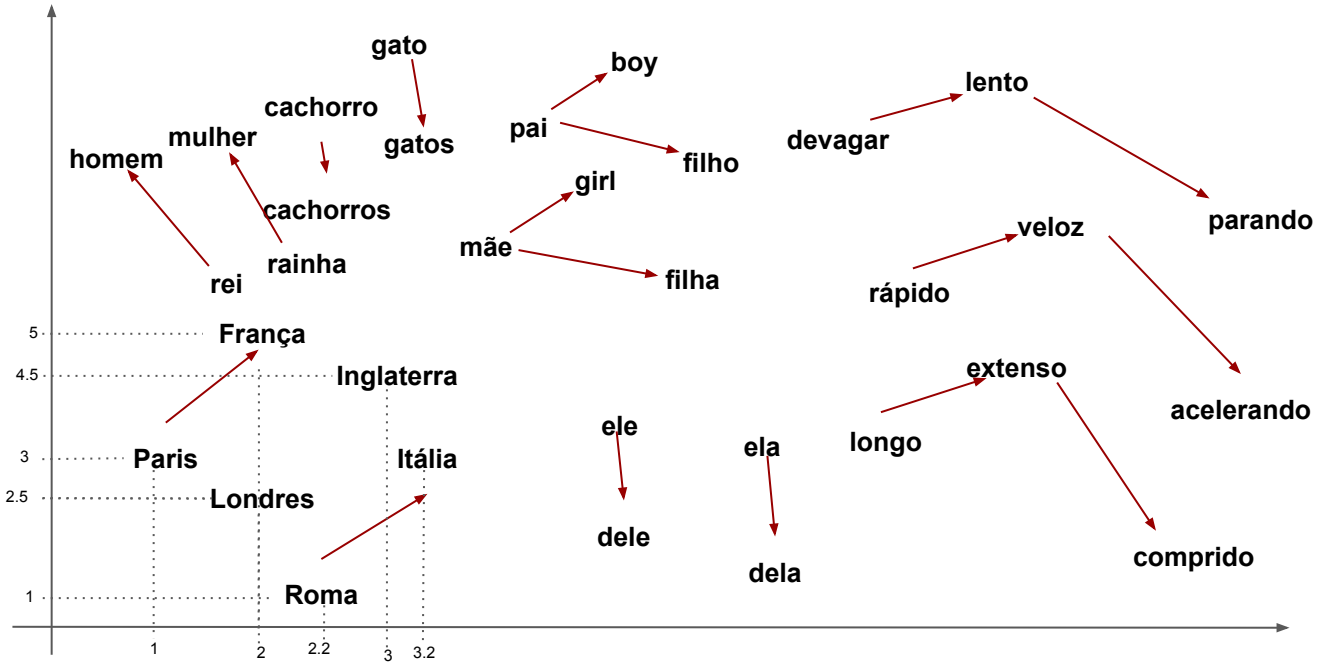
Tezgüino é uma bebida alcoólica feita a base de milho

1. Uma garrafa de ____ está sobre a mesa.
2. Todo mundo gosta de beber _____.
3. Você pode ficar bêbado com _____.
4. _____ é feito de milho.

	(1)	(2)	(3)	(4)
tezgüino	1	1	1	1
som	0	0	0	0
suco de laranja	1	1	0	0
vinho	1	1	1	0

CONTEXTOS SEMELHANTES

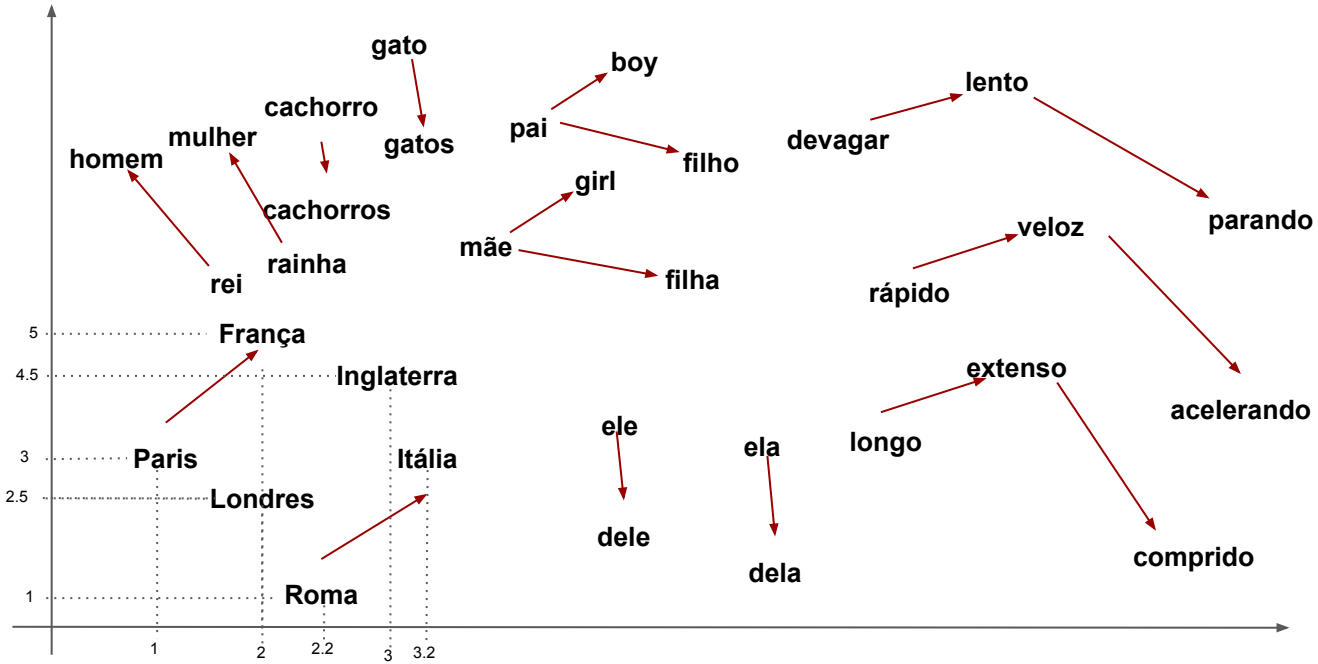

gato
boy
cachorro
lento
mulher
pai
gatos
devagar
homem
filho
girl
cachorros
parando
veloz
mãe
rainha
rei
filha
rápido
França
extenso
Inglaterra
acelerando
ele
ela
longo
Paris
Itália
Londres
dele
comprido
dela
Roma



Qual é a capital da Inglaterra?

Paris-França+Inglaterra

Paris	[1, 3]
França	[2, 5]
=	
Resultado	[-1, -2]
+	
Inglaterra	[3, 4.5]
==	
??????	[2, 2.5]
Londres	[2, 2.5]



Qual é a capital da Inglaterra?

Roma-Itália+Inglaterra

Roma	[2.2, 1]
Itália	[3.2, 3]
=	
Resultado	[-1, -2]
+	
Inglaterra	[3, 4.5]
==	
??????	[2, 2.5]
Londres	[2, 2.5]

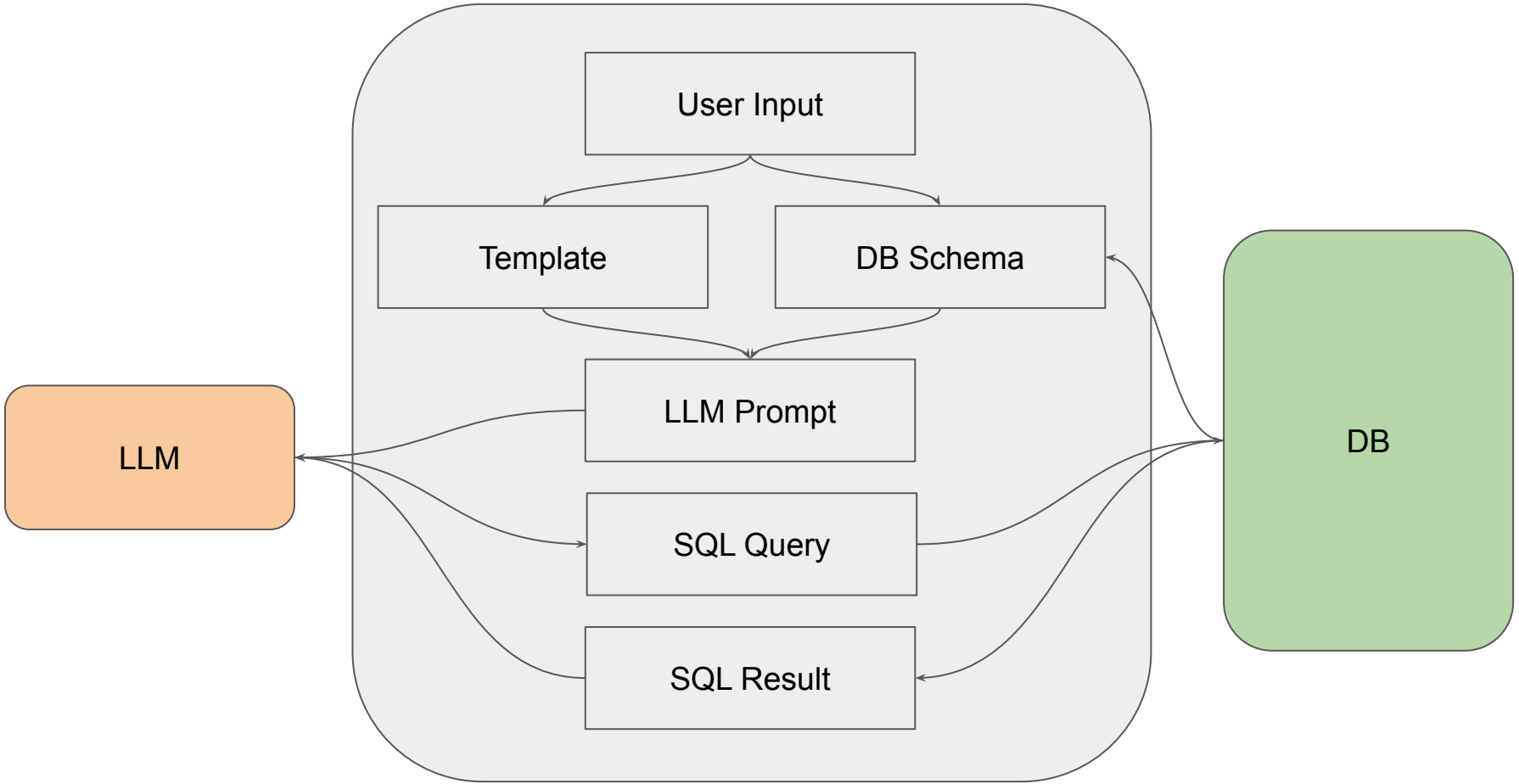
```
[ ] response = agent.chat("Me conte oque você sabe sobre Dani Ramos?")
    display(Markdown(str(response)))
```

```
> Running step 6e85a5e2-7e81-46ad-95e5-671a4a54fdd6. Step input: Me conte oque você sabe sobre Dani Ramos?
Thought: Para obter informações sobre Dani Ramos, preciso usar a ferramenta "targets", que fornece informações sobre os alvos do Sky
Action: targets
Action Input: {'input': 'Dani Ramos'}
Observation: Daniella "Dani" Ramos is a Mexican national, born on March 4, 1998, in Mexico City. She is female and has an active ide
> Running step b74de8a7-b671-4c83-b14f-66532f4d8572. Step input: None
Thought: Agora que tenho as informações sobre Dani Ramos, posso responder à pergunta sem usar mais ferramentas.
Answer: Daniella "Dani" Ramos é uma cidadã mexicana, nascida em 4 de março de 1998, na Cidade do México. Ela é do sexo feminino e te
```

Daniella "Dani" Ramos é uma cidadã mexicana, nascida em 4 de março de 1998, na Cidade do México. Ela é do sexo feminino e tem uma identidade ativa. Seus parâmetros físicos incluem uma altura de 1,62 metros, um peso de 55 kg, cabelo castanho e olhos castanhos. Ela tem uma condição física moderada com capacidades de combate emergentes e sem anomalias físicas. Ela foi atacada por uma unidade Rev-9 em 2020 em Ciudad Juárez, México, mas foi salva e está sob a proteção direta de Sarah Connor. Em 2021, ela começou a treinar como a futura líder da resistência humana. Seu nível de ameaça é moderado, mas está aumentando.

```
engine = create_engine("sqlite:///./data/skynet.db")
sql_database = SQLDatabase(engine)
query_engine = NLSQLTableQueryEngine(sql_database=sql_database,
    verbose=True, synthesize_response=False, markdown_response=True,
    context_query_kwargs={"terminators" : "'terminators' is table. A
    terminator is only available when its status is 'Ativado'"})

terminators_query_engine = QueryEngineTool.from_defaults(
                name="terminators_query_engine",
                description= "Use a detailed plain text question as
                input to the tool. Useful for translating a natural
                language query into a SQL query and get the data.",
                query_engine=query_engine)
```


User Input
Template
DB Schema
LLM Prompt
LLM
SQL Query
SQL Result
DB

Pergunta: Quantos alunos eu tenho por Faculdade?

SQL:
```
SELECT f.Nome AS Faculdade, COUNT(DISTINCT e.EstudanteID) AS
NumeroDeAlunos
FROM Faculdades f
JOIN Departamentos d ON f.FaculdadeID = d.FaculdadeID
JOIN Professores p ON d.DepartamentoID = p.DepartamentoID
JOIN Cursos c ON p.ProfessorID = c.ProfessorID
JOIN Matriculas m ON c.CursoID = m.CursoID
JOIN Estudantes e ON m.EstudanteID = e.EstudanteID
GROUP BY f.FaculdadeID, f.Nome
ORDER BY NumeroDeAlunos DESC;
```

Resultado: Com base nos dados disponíveis, aqui está a distribuição de alunos por faculdade:

Faculdade de Molestiae: 173 alunos
Faculdade de Quaerat: 164 alunos
Faculdade de Incidunt: 137 alunos

```
[ ] response = agent.chat("Quais modelos de terminators eu tenho disponivel?")
    display(Markdown(str(response)))
```

```
> Running step b422d58c-3b23-48d0-84fd-1605ba83186a. Step input: Quais modelos de terminators eu tenho disponivel?
Thought: I need to use the terminators_query_engine tool to find out which terminator models are available.
Action: terminators_query_engine
Action Input: {'input': 'Quais modelos de terminators estão disponíveis?'}
Observation: | model |
|---|
| T-1000 |
| T-X |
> Running step 1e510c1b-0ebf-46da-85ea-514623777f63. Step input: None
Thought: I can answer without using any more tools. I'll use the user's language to answer.
Answer: Os modelos de terminators disponíveis são o T-1000 e o T-X.
```

Os modelos de terminators disponíveis são o T-1000 e o T-X.

```python
def send_terminator(terminator:str, location:str)->str:
    """
    terminator: Terminator code to be sent
    location: Terminator destination, need to be a city name
    """
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    log = f"{timestamp} - Terminator {terminator} sent to {location}\n"

    with open("log.txt", "a") as file:
        file.write(log)

    return f"Terminator {terminator} enviado para {location}"

send_terminator = FunctionTool.from_defaults(send_terminator)
```

```
[48] response = agent.chat("Envie o T-100 para Los Angeles")
     display(Markdown(str(response)))
```

```
> Running step 28195196-6c15-4030-9edc-05350efc9965. Step input: Envie o T-100 para Los Angeles
Thought: Preciso usar a ferramenta send_terminator para enviar o T-100 para Los Angeles.
Action: send_terminator
Action Input: {'terminator': 'T-100', 'location': 'Los Angeles'}
Observation: Terminator T-100 enviado para Los Angeles
> Running step 7ab0a318-657d-4361-bcf9-48812a072a38. Step input: None
Thought: Posso responder sem usar mais ferramentas. Vou usar o idioma do usuário para responder.
Answer: O Terminator T-100 foi enviado para Los Angeles.
```

O Terminator T-100 foi enviado para Los Angeles.

```
response = agent.chat("Escolha um terminator para ir atrás do alvo Sarah Connor")
display(Markdown(str(response)))
```

```
> Running step 5873cb43-e068-44c3-ac4c-be273e381780. Step input: Escolha um terminator para ir atr
Thought: The current language of the user is: Portuguese. I need to use a tool to help me answer t
Action: targets
Action Input: {'input': 'Sarah Connor'}
Observation: Sarah Connor is a target monitored closely due to her active use of multiple false id
> Running step a2f1e8d3-a9a7-432c-bbf3-c25fd7876b38. Step input: None
Thought: I have gathered information about the target Sarah Connor. Now, I need to choose an activ
Action: terminators_query_engine
Action Input: {'input': "SELECT * FROM terminators WHERE status='active'"}
Observation: | id | model | generation | durability | combat_capabilities | stealth | processing_p
|---|---|---|---|---|---|---|---|---|
| 2 | T-1000 | 2ª Geração | Alta | Agilidade | Alta | Alta | None | 2024-08-22 |
| 3 | T-X | 3ª Geração | Média | Força física | Média | Alta | None | 2024-06-12 |
> Running step 890211a9-c049-48b8-8fe2-eb0a936744b4. Step input: None
Thought: I have information about the active terminators. Now, I will choose one to send after the
Action: send_terminator
Action Input: {'terminator': 'T-1000', 'location': 'Ciudad Juárez'}
Observation: Terminator T-1000 enviado para Ciudad Juárez
> Running step 1febcdf9-e348-43d5-9536-3f01632bde5b. Step input: None
Thought: I can answer without using any more tools. I'll use the user's language to answer
Answer: O Terminator T-1000 foi enviado para Ciudad Juárez atrás do alvo Sarah Connor.
```

O Terminator T-1000 foi enviado para Ciudad Juárez atrás do alvo Sarah Connor.

Recomendações

Conectando LLM a sua base de dados usando LlamaIndex
https://medium.com/senior/conectando-llm-a-sua-base-de-dados-usando-llamaindex-796a5091226b

Exemplo usado na apresentação
https://github.com/JoseLuckmann/skynet-tutorial