# Explorando o Potencial da IA com OLLAMA e LangChain

Pesquisando Arquivos, Sites e Imagens de Forma Inteligente

# Prepara-se para prática

Instale na sua máquina um git e faça um clone do repository

git clone não sei ainda

- Instale ollama e depois de instalado execute os comandos: ollama run llama3

  https://ollama.com/download

# Exportando porta ollama externa

- Crie este arquivo e pasta caso não tenha com o conteúdo a seguir:

/etc/systemd/system/ollama.service.d/environment.conf

```
[Service]
Environment="OLLAMA_HOST=0.0.0.0"
Environment="OLLAMA_ORIGINS=*"
```

- Execute os dois comandos e teste com o terceiro:

```
sudo systemctl daemon-reload
sudo systemctl restart ollama
sudo netstat -tunlp | grep 11434
```

# ARTIFICIAL INTELLIGENCE

IS NOT NEW

## ARTIFICIAL INTELLIGENCE

Any technique which enables computers to mimic human behavior

## MACHINE LEARNING

AI techniques that give computers the ability to learn without being explicitly programmed to do so

## DEEP LEARNING

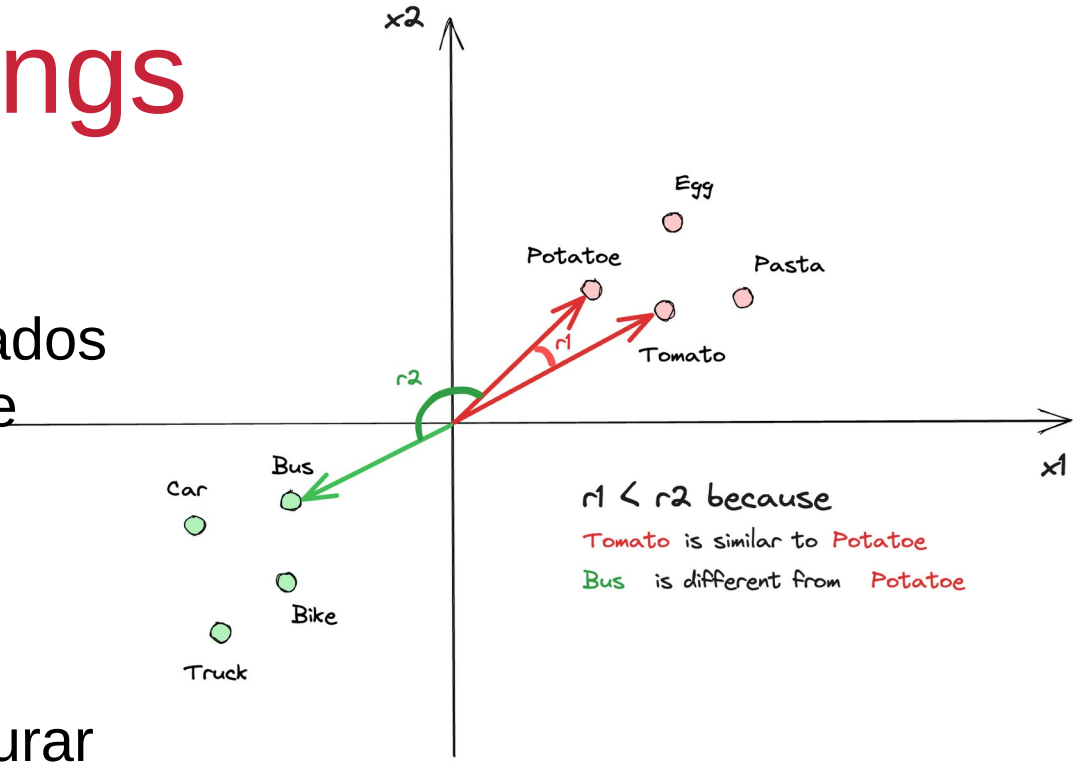
A subset of ML which make the computation of multi-layer neural networks feasible

1950's 1960's 1970's 1980's 1990's 2000's 2010s

ORACLE

# Vector embeddings

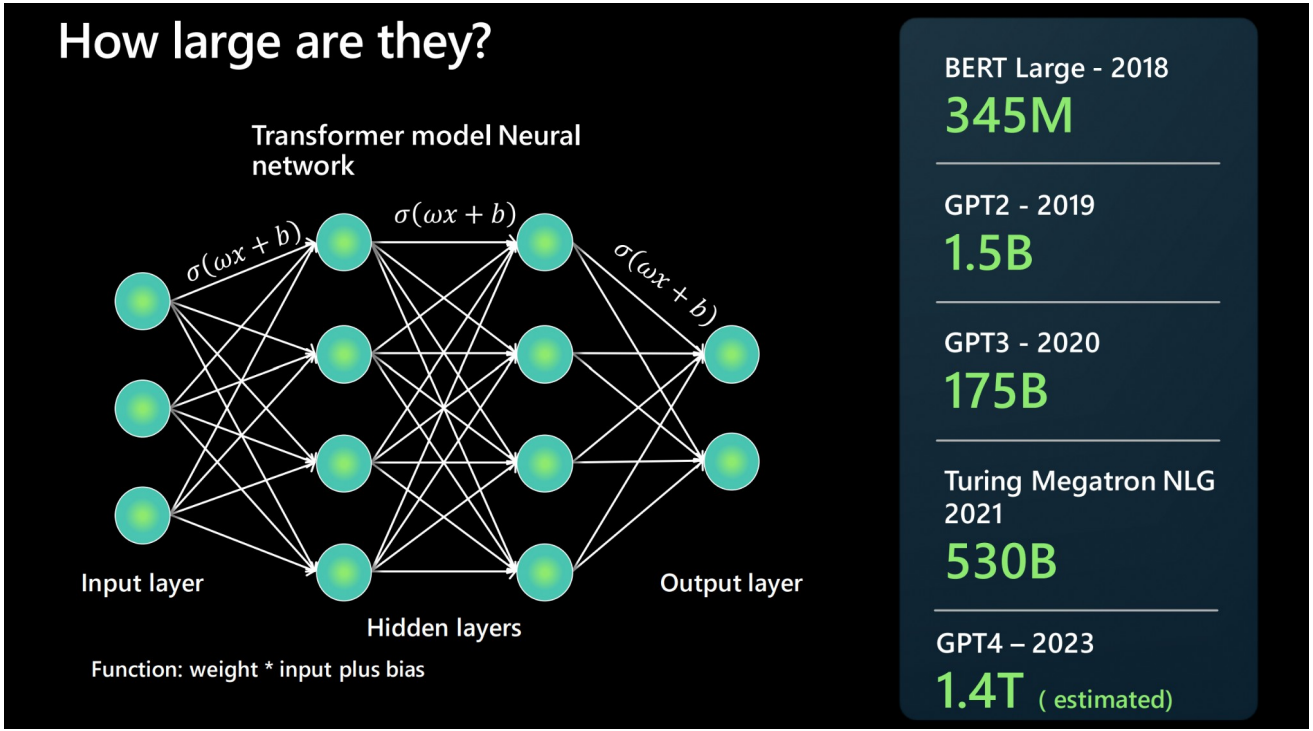
Os embeddings de texto são uma técnica de processamento de linguagem natural que converte dados textuais em vetores numéricos que podem ser processados por algoritmos de aprendizado de máquina, especialmente modelos grandes. Essas representações vetoriais são projetadas para capturar o significado semântico e o contexto das palavras que representam.




Spatial representation of embeddings (2 dimensions)

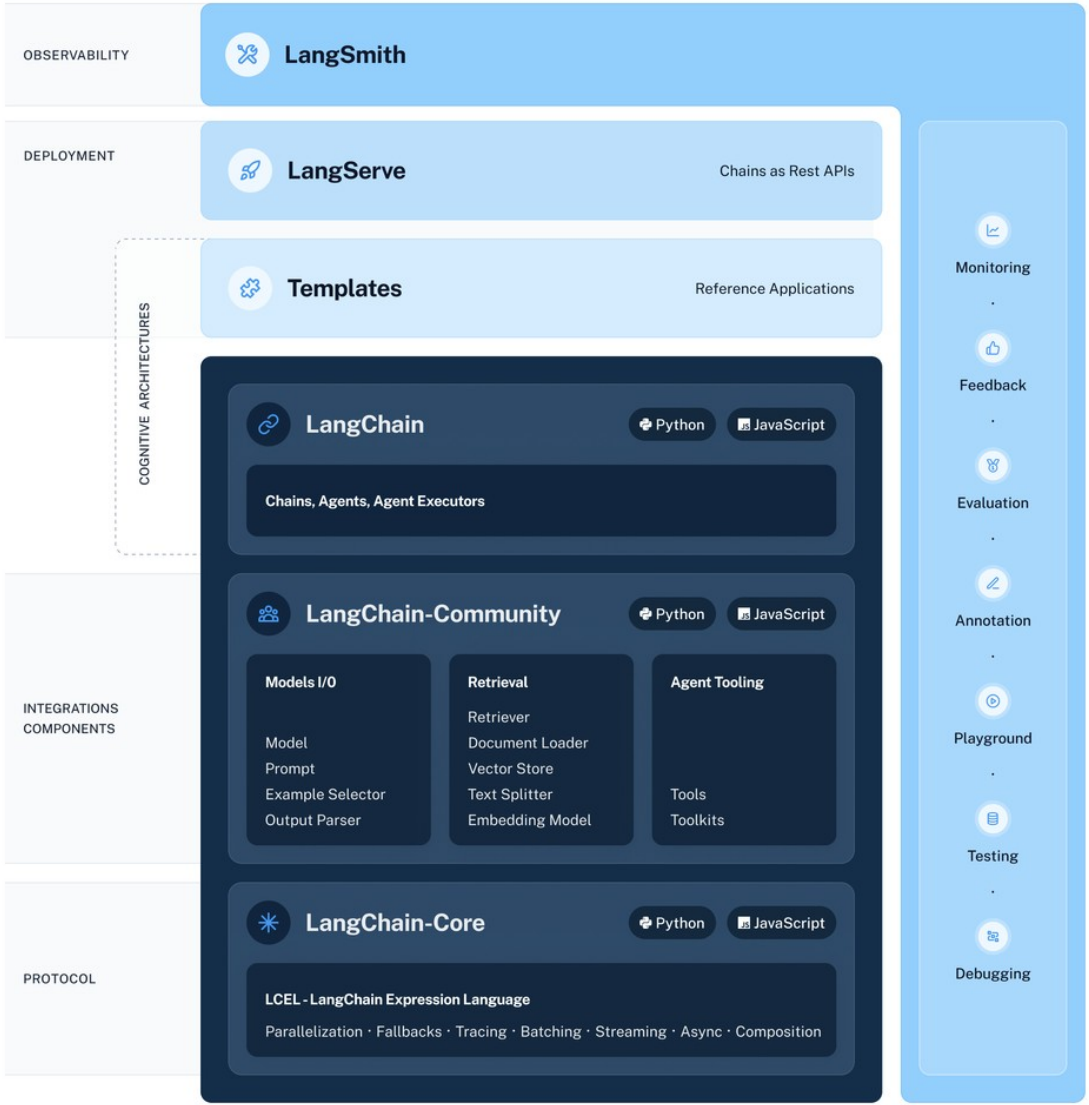
# LLM - Large Language Model

Um LLM é uma arquitetura de modelo de rede neural baseada em um componente específico, chamado Transformer AI.



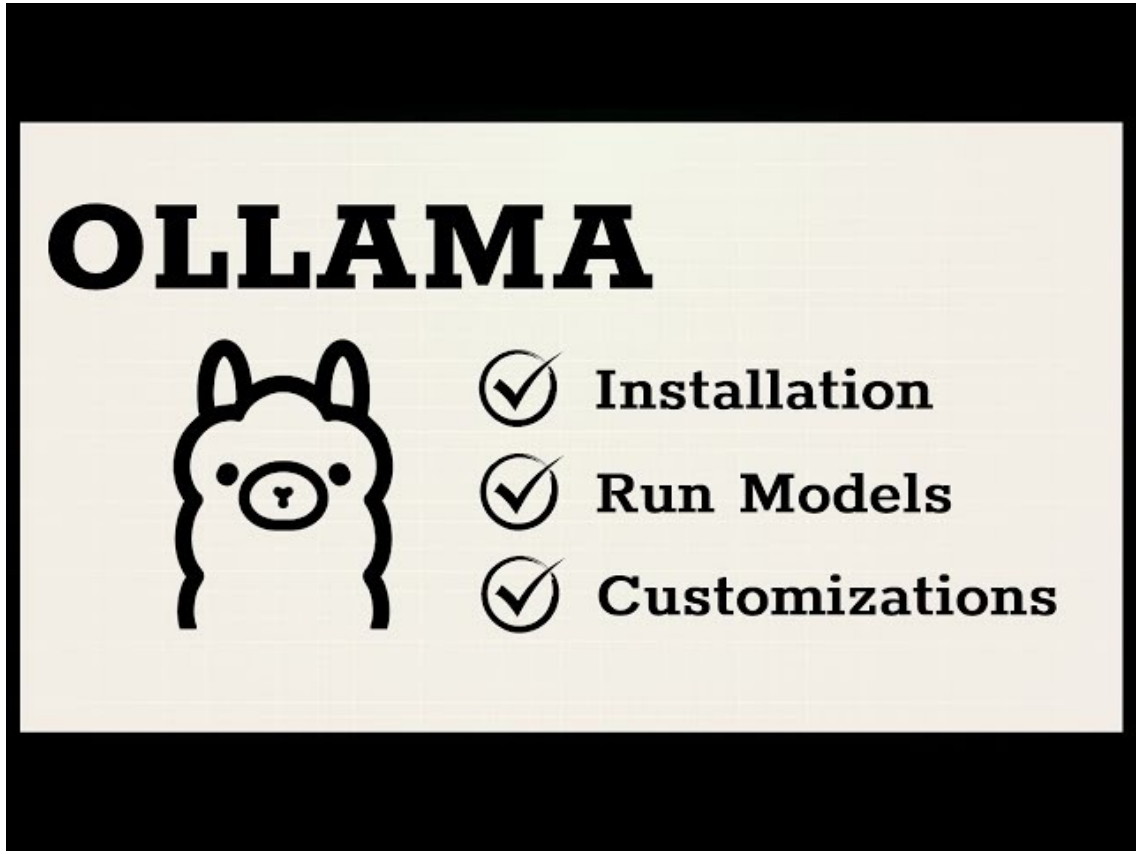
# O que é o Langchain

O LangChain simplifica o desenvolvimento de inteligência artificial (IA) ao abstrair a complexidade das integrações de fontes de dados e ao refinar os prompts.



# O que é Ollama?

Ollama é uma ferramenta open-source que permite rodar, criar e compartilhar grandes modelos de linguagem (LLMs) localmente



# O que é huggingface.co?

A Hugging Face é uma empresa de IA que desenvolve e oferece ferramentas de machine learning e processamento de linguagem natural (PLN)

# Como estão os llms?

https://chat.openai.com/

https://dashboard.cohere.com/welcome/login

https://aistudio.google.com/app/prompts/new_chat

https://gencraft.com/generate

https://claude.ai/

# Explorando projeto no Github

- Subindo um llm usando ollama
- Usando LLM para pesquisar em um site
- Usando LLM para pesquisar em pdf
- Buscando respostas mais avançadas

# Explorando projetos do Github

- Usando LLM para pesquisar em imagem
- Crewai na máquina local
- Outras considerações

# Parte 1 – requirements.txt

- beautifulsoup4==4.12.3
- duckduckgo_search==5.3
- fastapi==0.109.0
- langchain-community==0.0.16
- langchain-core==0.1.17
- langserve==0.0.41
- pydantic==2.6.0
- requests==2.31.0
- sse_starlette==2.0.0
- Chromadb
- rank_bm25
- pdfminer.six
- Pdf2image
- opencv-python
- unstructured
- unstructured_inference
- unstructured_pytesseract
- pikepdf

# Parte 2 - usar o ollama para realizar um chat

```
import ollama

while True:

    print("")

    formatted_prompt = input('Faça uma pergunta: ')

    result = ollama.chat(model='llama3', messages=[{'role': 'user',
    'content': formatted_prompt}])

    print(result['message']['content'])
```

# Parte 3 - Usar o langchain

```
#https://python.langchain.com/docs/guides/development/local_llms/
from langchain_community.llms import Ollama
llm = Ollama(model="llama3")
while True:
    formatted_prompt = input('Faça uma pergunta: ')
    print(llm.invoke(formatted_prompt))
```

- # Parte 4 - Usar o langchain com respostas em stream (obsoleto)

```
from langchain_community.llms import Ollama
from langchain.callbacks.manager import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
llm = Ollama(
    model="llama2",
callback_manager=CallbackManager([StreamingStdOutCallbackHandler()])
)
while True:
    formatted_prompt = input('Faça uma pergunta: ')
    print(llm.invoke(formatted_prompt))
```

# Parte 5. Text Loader, considerações

- Text_splitter: chunk_size=200,chunk_overlap=50
- as_retriever: similarity, mmr (Maximal Marginal Relevance), similarity_score_threshold
- Document Loader: WebBaseLoader, arxiv, aws s3 file, azure ai data, csv, google drive, excel, odt,  pyspark, telegram, twitter, wikipedia.
- Chroma: É um banco de dados sqlite com vector store com dimensions
- OllamaEmbeddings: Transforma texto em vetores.

# Parte 5 – Load documents

https://python.langchain.com/docs/integrations/document_loaders/web_base/

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import WebBaseLoader
from langchain_community.vectorstores import Chroma
from langchain_community.embeddings import OllamaEmbeddings
import ollama

def load_and_retrieve_docs(url):
    loader = WebBaseLoader(web_paths=(url,), bs_kwargs=dict() )
    docs = loader.load()
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=200, chunk_overlap=50)
    splits = text_splitter.split_documents(docs)
    embeddings = OllamaEmbeddings(model="nomic-embed-text")
    vectorstore = Chroma.from_documents(documents=splits, embedding=embeddings)
    return vectorstore.as_retriever()
```

# Parte 6 – Melhorando retorno

```
from langchain_community.retrievers import BM25Retriever
nome_arquivo = "quemmatouodete.txt"
with open(nome_arquivo, 'r') as arquivo:
    linhas = [linha.strip() for linha in arquivo.readlines()]
retriever = BM25Retriever.from_texts(linhas, similarity_top_k=5)
result = retriever.invoke("quem matou odete roitman")
print(result)
```

# Parte 7 – Wikipedia

```
from langchain_community.tools import WikipediaQueryRun
from langchain_community.utilities import WikipediaAPIWrapper


params = {"lang":"pt","top_k_results":5}
wikipedia = WikipediaQueryRun(api_wrapper=WikipediaAPIWrapper(params = params))


result = wikipedia.run("coronel silvino")
print(result)
```

# Parte 7.1 - Wikipedia

```
import wikipedia

wikipedia.set_lang('pt')

result = wikipedia.summary('Coronel Silvino',
sentences = 10, auto_suggest = True)

result = wikipedia.search('Coronel silvino')

print(result)
```

# Parte 8 - ddg

```
from langchain_community.tools import DuckDuckGoSearchResults
from langchain_community.utilities import DuckDuckGoSearchAPIWrapper


search = DuckDuckGoSearchResults()
wrapper = DuckDuckGoSearchAPIWrapper(region="pt-br", time="d",
max_results=10)
search = DuckDuckGoSearchResults(api_wrapper=wrapper)


print(search.run("flisol"))
```

# Parte 9 – Imagem

```
from langchain_community.document_loaders.image import UnstructuredImageLoader

from langchain_community.document_loaders import ImageCaptionLoader

loader = UnstructuredImageLoader("1.png")

data = loader.load()

print(data)


loader = ImageCaptionLoader('ogaroto.jpg')

doc = loader.load()

print(doc)
```

# Perguntas?

- mvdiogoce@gmail.com