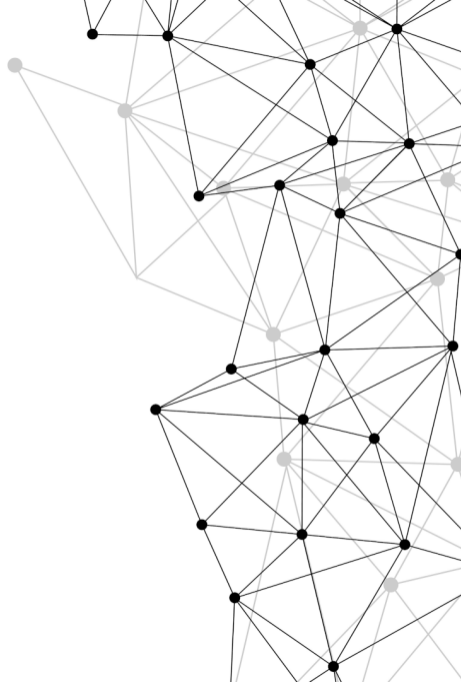




# Workshop - Dados e treinamento

Estrutura, recursos e análise de dados

Lucas Migliorin da Rosa



# Table of Contents

## 1 Quem sou?

► Quem sou?

► Parte I

► Part II

► Part III

# Um pouco sobre mim

## 1 Quem sou?

### Lucas Migliorin

Desenvolvedor de Inteligência Artificial

#### Resumo

- Atuo no desenvolvimento de soluções de IA, com experiência prática em treino e validação de modelos e na construção de pipelines de Deep Learning.
- Interesse-me por IA no geral, algoritmos matemáticos e aplicações.

#### Áreas

- Machine Learning / Deep Learning
- LLMs e aplicações
- Ciência de Dados

#### Stacks

- MLOps
- Pytorch / Tensorflow
- Sklearn
- Pandas e Numpy



# Table of Contents

2 Parte I

▶ Quem sou?

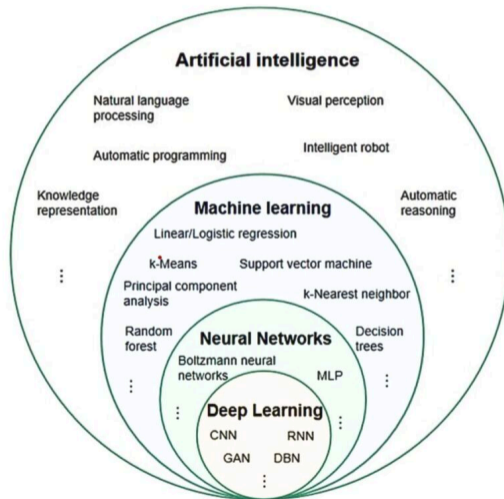
▶ Parte I

▶ Part II

▶ Part III

# O conjunto da IA

## 2 Parte I



# Ferramentas

## 2 Parte I



**Figura:** Bibliotecas mais usadas para exploração, validação e treinamento.



# Infraestrutura

## 2 Parte I

- Machine Learning (Algoritmos clássicos)
  - Menos recursos
  - Tendem a ser mais leves
- Deep Learning (Baseados em redes neurais)
  - Depende da escolha do algoritmo
  - Mais pesados e robustos
- Large Language Model (LLMs)
  - Largas consequentemente pesadas
  - Maior poder computacional (Depende)

# Algoritmos famosos

## 2 Parte I

Categoria	Algoritmo
Machine Learning	Regressão Linear
	Regressão Logística
	K-Nearest Neighbors (KNN)
	Support Vector Machine (SVM)
	Decision Tree
	Random Forest
	Gradient Boosting (XGBoost, LightGBM, CatBoost)
Deep Learning	Naive Bayes
	Perceptron Multicamadas (MLP)
	Convolutional Neural Networks (CNN)
	Recurrent Neural Networks (RNN)
	LSTM (Long Short-Term Memory)
	GRU (Gated Recurrent Unit)
LLMs	Autoencoders
	Transformer
	BERT
	GPT (Generative Pre-trained Transformer)
	T5 (Text-to-Text Transfer Transformer)
	LLaMA
	Mistral

**Tabela:** Separação de redes e algoritmos conhecidos por categoria



# Table of Contents

3 Part II

▶ Quem sou?

▶ Parte I

▶ **Part II**

▶ Part III

# Modelos de Linguagem Natural (LLM)

## 3 Part II

- Baseados em redes Transformers<sup>1</sup>
- Os “b” são quantidades de parâmetros
- Extensão GGUF (Somente inferência)
- Quantização
  - Conversão de *float32* a *int8*
  - (↓) Menos espaço
  - (↓) Menos precisão

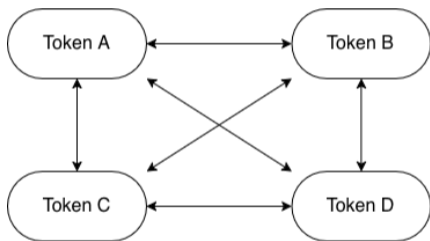
FP16 ↓ BF16 ↓ INT8 ↓ INT4 ↓ FP8 ↓ INT2

---

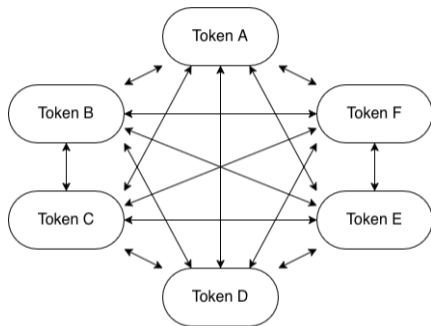
<sup>1</sup><https://arxiv.org/abs/1706.03762>

# Tokens e influência

## 3 Part II



(a) LLM com "4b"



(b) LLM com "6b"

**Figura:** Relações entre os tokens conforme escolhesse uma rede com mais parâmetros.

# Rodando localmente

## 3 Part II

- Requisitos recomendados
  - OS: Linux ou WLS2 (Windows)
  - GPU: mínimo 8GB VRAM (Não é obrigatório)
  - Docker
  - Driver CUDA
- Perguntas que há de surgir
  - Por quê Linux ?
  - Por quê GPU não é obrigatória ?
  - Posso rodar sem usar Docker ?

# Llama.cpp Anywhere Docker

## 3 Part II

```
services:
  llamacpp-server:
    image: ghcr.io/ggml-org/llama.cpp:server-cuda
    container_name: llama_cpp_container
    ports:
      - 8080:8080
    volumes:
      - /home/ti/models/gguf-models/:/models
    environment:
      # alternatively, you can use "LLAMA_ARG_MODEL_URL" to download the model
      LLAMA_ARG_MODEL: /models/Meta-Llama-3.1-8B-Instruct-Q4_K_M.gguf
      LLAMA_ARG_N_GPU_LAYERS: 55
      LLAMA_ARG_NO_WEBUI: 0
      LLAMA_ARG_PORT: 8080
      LLAMA_ARG_HOST: 0.0.0.0
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
```

# Compatibilidades do llama.cpp

## 3 Part II

- API OpenAI-compatible (chat/completions/embeddings)
- Compatível com SDK oficial `openai` (Python/JS)
- Compatível com agentes
- Suporte a clientes HTTP padrão (curl, requests, axios)
- WebUI e UIs compatíveis (OpenWebUI, LM Studio via backend compatível)

# Entendendo o Prompt

## 3 Part II

### Prompt

# Contexto

Você é um controlador de comandos de um chat.

# Objetivo

Seu objetivo é mandar comandos conforme o contexto da conversa. Seu único objetivo é enviar os comandos disponíveis.

# Regras

Comandos disponíveis:

- comando01() -> Quando a usuária pedir para fazer um agendamento

...

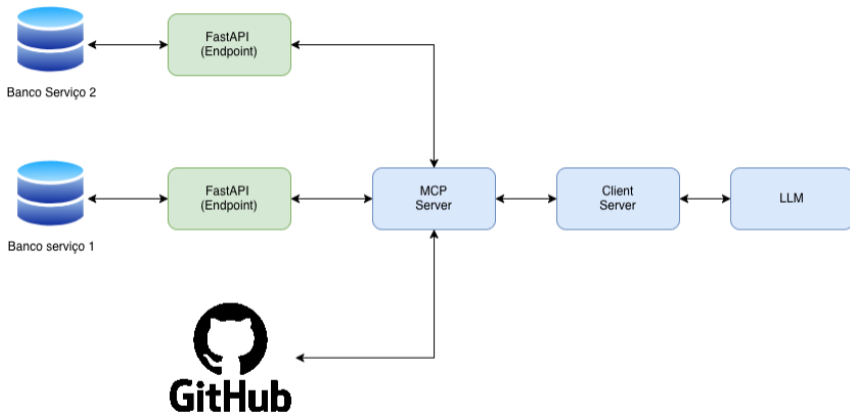
# Resposta

Suas respostas serão sempre comandos da lista de comandos disponíveis

- A saída é sempre uma String - Verificar a saída

# MCP Server

## 3 Part II



**Figura:** Arquitetura de integração de um server MCP com LLM

# Aplicando ao MCP

## 3 Part II

### Prompt

# Contexto

Você é um agente que pode utilizar ferramentas externas via MCP.

# Objetivo

Analisar a conversa e chamar a ferramenta apropriada quando necessário.

# Regras

Ferramentas disponíveis:

- agendar\_consulta(data:str, tipo:int) -> Agenda uma consulta médica

...

# Resposta

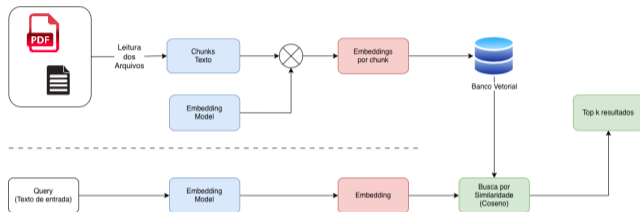
A resposta deve conter apenas a chamada estruturada da ferramenta no formato JSON:

```
{  
  "tool_call": "agendar_consulta",  
  "arguments": { ... }  
}
```

- A saída é sempre uma String - Verificar a saída

# Fluxo RAG

## 3 Part II



- Arquivos de texto em chunks
  - Chunk size: 256 512 (normalmente)
  - Overlap: 10% (normalmente)
- Language Model mais simples
- Banco vetorizado

# Table of Contents

4 Part III

▶ Quem sou?

▶ Parte I

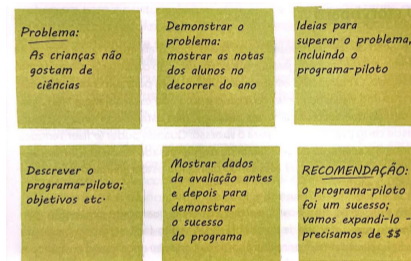
▶ Part II

▶ Part III

# Início da ideia

## 4 Part III

1. Pense antes o que você quer analisar
2. Comece organizando suas ideias com um *storybord* (Opcional)
3. Não se apegar ao seu trabalho



**Figura:** Exemplo de Storyboard do livro Storytelling com Dados

# Análise Exploratória de Dados

## 4 Part III

- Métodos tradicionais
  - Média
  - Mediana
  - Desvio padrão
- Visualizações e Outliers
  - Boxplot
  - Scatterplot
  - Barras
  - Histogramas
- Correlação
  - Matrix Correlação

# Fluxo Geral

## 4 Part III



- Pré-processamento

- Normalização
- Remoção Outliers
- Fill NAN (Muitos métodos)

- Validação

- Reg: MSE, MAE, RMSE,  $R^2$
- Cls: Recall, ACC, F1, Mat. confusão



*Q&A*

**Obrigado!**