

# **Inteligência Artificial Generativa: Parte 1: Introdução**

Thiago Brandenburg  
2025/1

# Apresentação

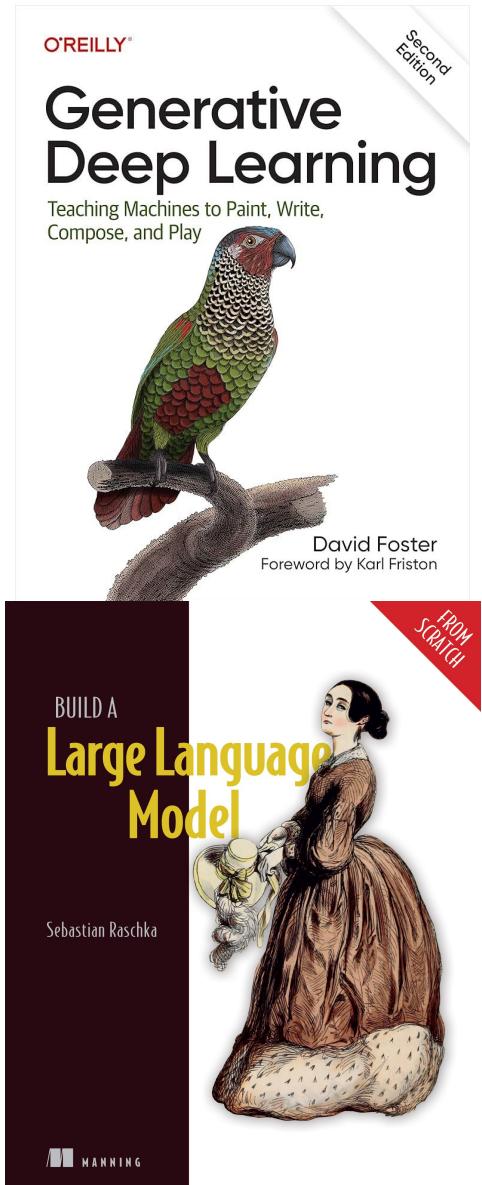
- Thiago Brandenburg
- Formado em Ciência da Computação na UDESC (2024)
- Aluno do Mestrado em Computação Aplicada da UDESC.
- Membro Laboratório de Inteligência Computacional (LABICOM).
- Atuo no projeto “Aprendizado de máquina para predição da taxa de corrosão em aços”
  - Desenvolvimento de modelos de predição de salinidade do ar.
  - Desenvolvimento de modelos de corrosão atmosférica.

# Informações gerais

- Cronograma
  - Aula 1: Introdução a modelos generativos.
  - Aulas 2 e 3: LLMs, Arquitetura Transformer.
  - Aulas 4 e 5: Geração de Imagens, Variational Autoencoders, modelos de difusão.
- Linguagem de Programação: Python
  - Pandas - Biblioteca de processamento de dados.
  - Matplotlib - Biblioteca de plotagem de dados.
  - Keras - Biblioteca para Redes Neurais Artificiais.
  - Transformer - Biblioteca do Hugging Face para LLMs.
- Ambiente: Google Colab (Jupyter Notebook).
- Método de avaliação: questionários + análise de código.

# Bibliografia

- Generative Deep Learning, 2<sup>a</sup> edição.
  - Livro base.
  - Contém repositório com códigos utilizados no livro.
  - Utilizem a 2<sup>a</sup> edição!!!
- Build a Large Language Model from Scratch:
  - Aprofunda a arquitetura de modelos de linguagem.
- Disponibilizarei um arquivo com referências aos demais materiais utilizados no desenvolvimento do conteúdo.



# Expectativas

- Projetei o conteúdo para ensinar como modelos generativos são construídos, **passando por cada parte de sua arquitetura.**
- O que esperar?
  - Arquiteturas de modelos generativos.
  - Aprofundamento em conceitos dos modelos **atuais** generativos.
  - Modelos de texto e imagem.
- O que **não** esperar?
  - Usos de gen AI (engenharia de prompt).
  - Outras mídias.

Arquitetura



Prompt



# ■ Contextualização

- O que vem a mente para o termo **GenAI**?

# ■ Contextualização

- O que vem a mente para o termo **GenAI**?
- Vamos começar pelos chatbots disponibilizados pelas grandes corporações de TI.
  - **ChatGPT** da OpenAI
  - **Gemini** da Google
  - **Meta AI** da Meta
  - **Deepseek**

## Text Generation



# Contextualização

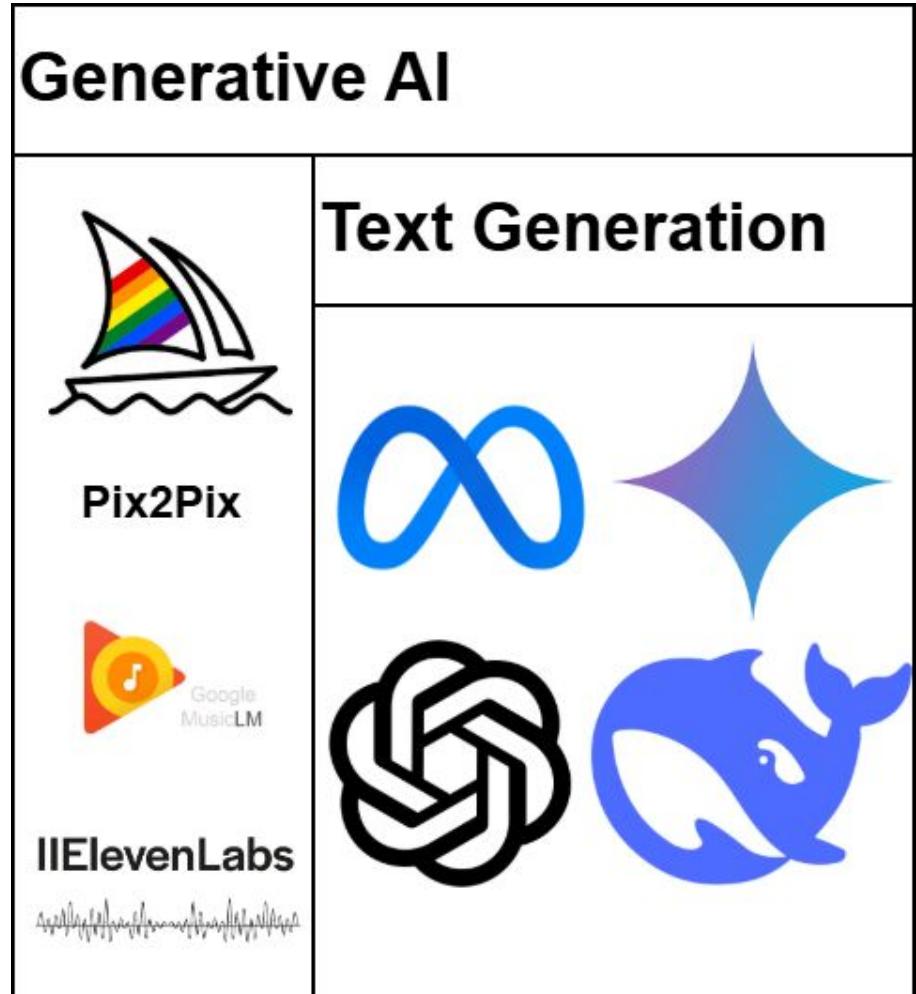
- Esses chatbots são modelos generativos de texto, que fazem parte de um grupo específico: **Grandes Modelos de Linguagem (*Large Language Models*, sigla LLM).**
- Os grande modelos de linguagem incluem outras funcionalidades, como tradução de texto, correção, identificação de linguagem imprópria, etc.
- Os LLMs, por sua vez, estão dentro da área de **Processamento de Linguagem Natural (PLN).**

## Text Generation



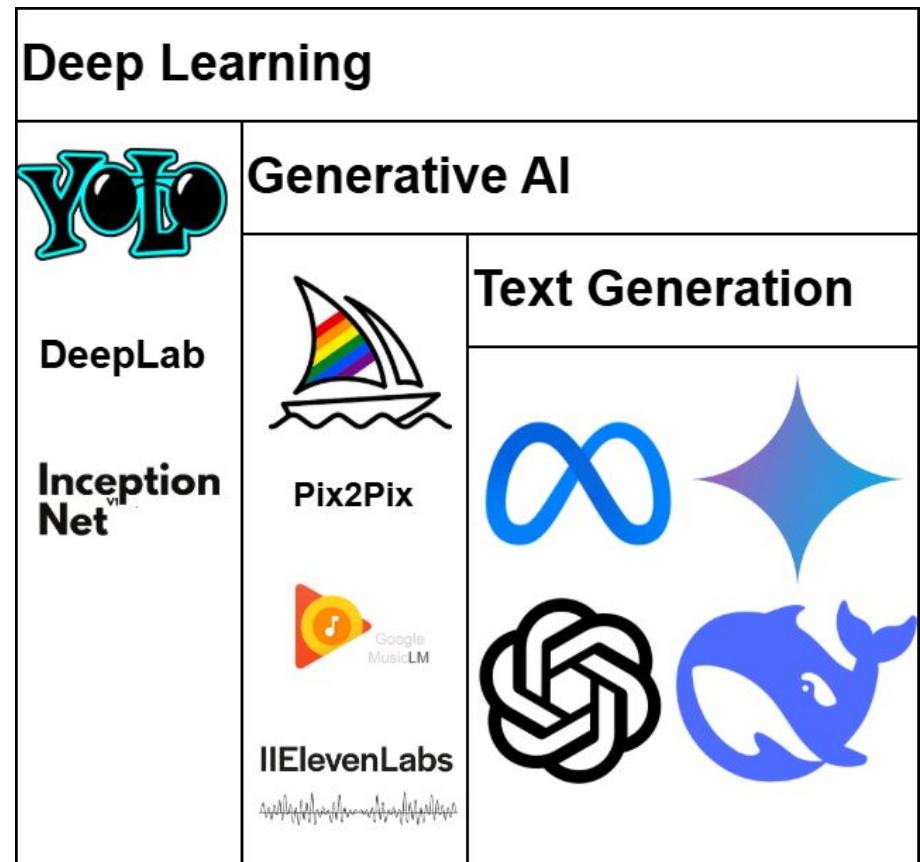
# Contextualização

- A geração de texto é apenas um tipo de GenAI, atualmente diferentes tipos de mídias que podem ser gerados:
- Referimos as mídias geradas por uma gen AI como **modalidade**.
- **Imagens**: Midjourney, DALL-E, Stable Diffusion, Pix2Pix.
- **Vídeo**: Sora.
- **Voz**: Speechify, ElevenLabs.
- **Música**: Soundraw, Google MusicLM.
- etc.



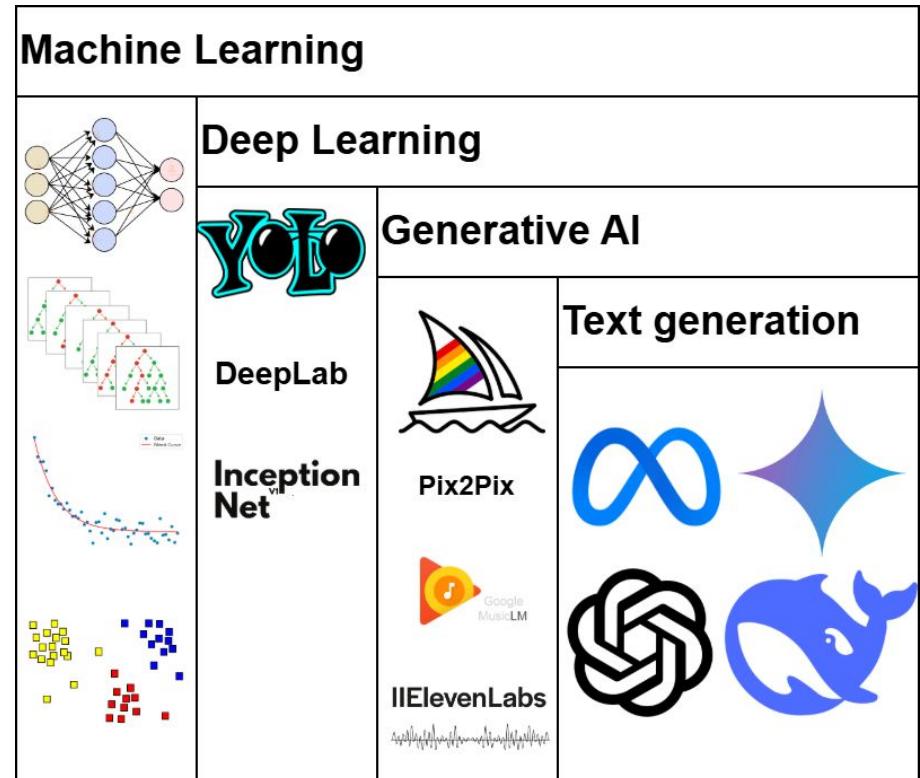
# Contextualização

- Atualmente, todos os modelos gerativos são baseados e **Redes Neurais Artificiais Profundas.**
  - Camadas de Perceptrons.
  - Milhares/Milhões de parâmetros a serem ajustados.
  - Volume grande de dados não estruturados.
  - Alto processamento para treinamento/execução.
- Outros modelos de deep learning incluem principalmente processamento de imagem.



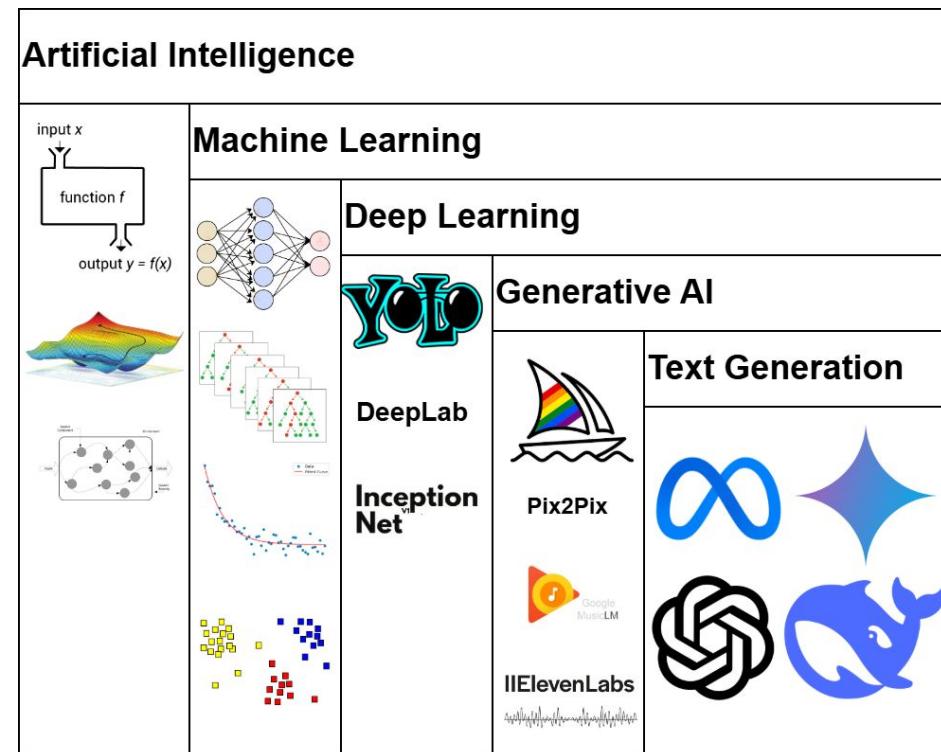
# Contextualização

- Redes Neurais Artificiais são um tipo de modelo de Machine Learning.
- Portanto, podemos questionar quais são as **características** de modelos ML que as gen AIs possuem:
  - Qual é o tipo de **aprendizado** utilizado?
  - Qual é a **métrica** utilizada para ajustar o modelo?
  - Como é organizada as **camadas** da ANN?
  - Existe alguma diferença para os modelos “tradicionais”?



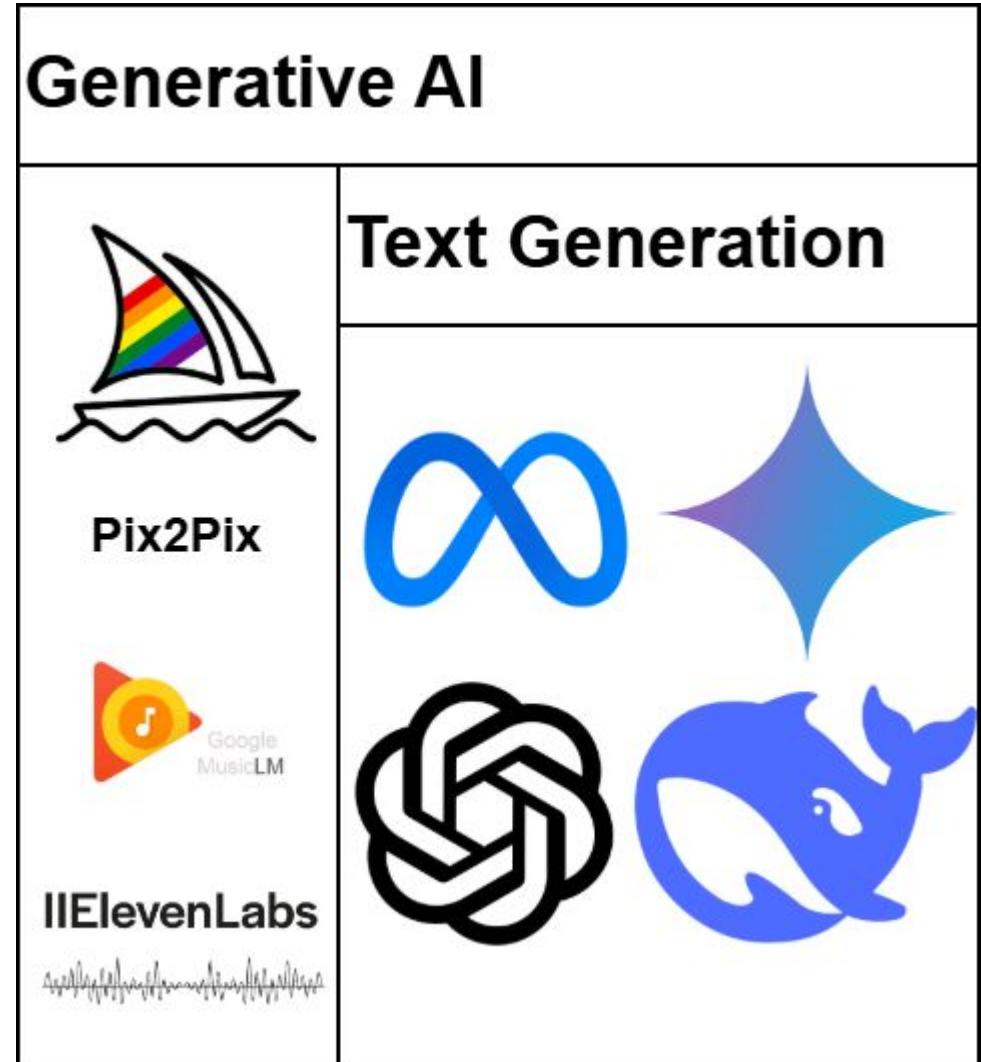
# Contextualização

- A Inteligência Artificial abriga não só Machine Learning, mas também otimização, tomada de decisão, simulação.
- Portanto, A **inteligência artificial generativa** consiste de modelos de **Machine Learning** construídos utilizando **redes neurais profundas**, para diferentes tipos de **modalidade**.



# Contextualização

- Nessas aulas, teremos foco em IA Generativa, com desenvolvimentos em **imagem e texto**.

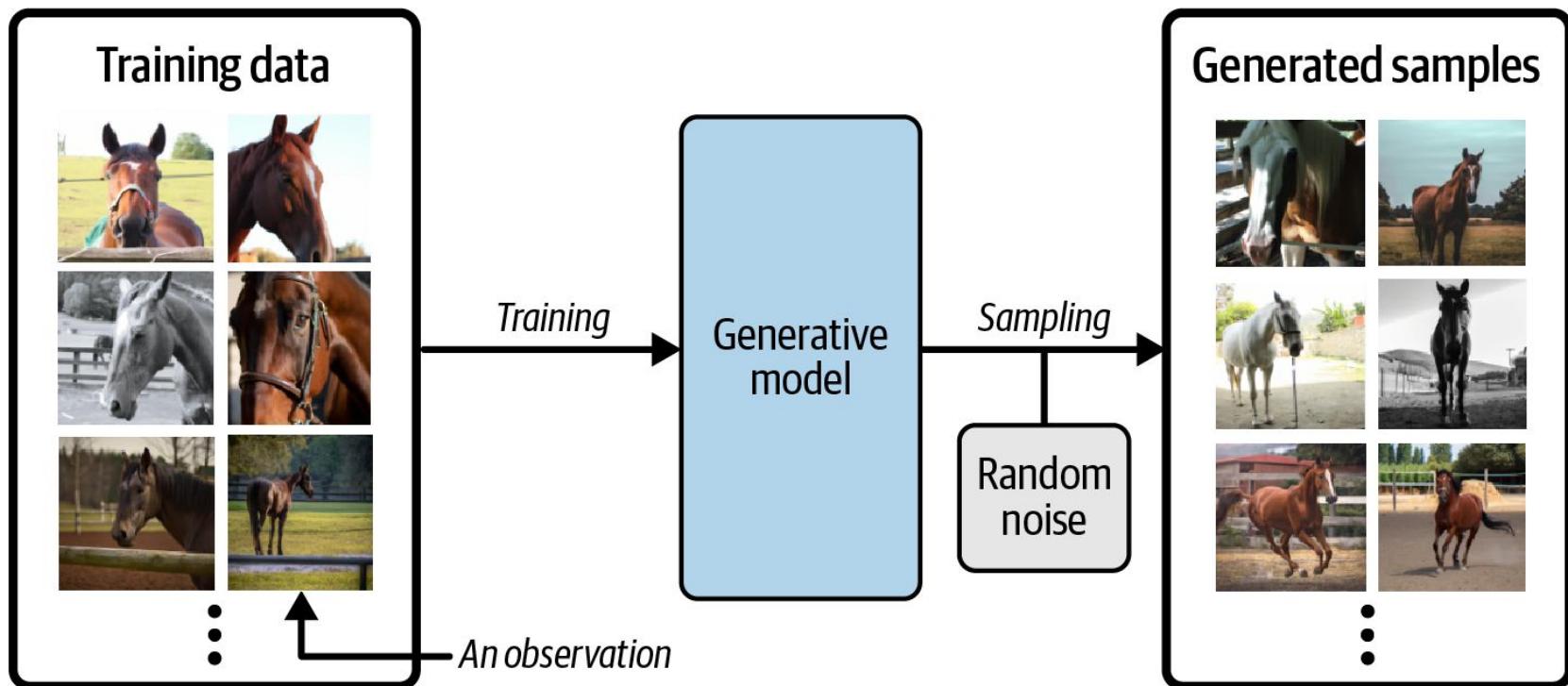


# O que é Inteligência artificial generativa?

- A IA generativa permite aos utilizadores gerar rapidamente novos conteúdos com base numa variedade de inputs. Os inputs e outputs para estes **modelos** podem incluir texto, imagens, sons, animação, modelos 3D ou outros tipos de dados. **NVIDIA**
- A IA generativa refere-se a **modelos** de aprendizagem profunda que podem gerar texto, imagens e outros conteúdos de alta qualidade com base nos dados em que foram treinados. **IBM**
- A inteligência artificial generativa se refere a **área** que estuda os **modelos generativos**.

# Definição

- A modelagem generativa é um ramo da área de machine learning que envolve a treinar de um **modelo** para produzir novos dados que sejam semelhantes ao do dataset fornecido (Foster, 2023).

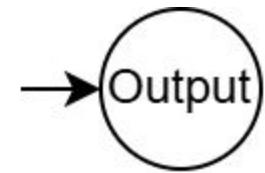
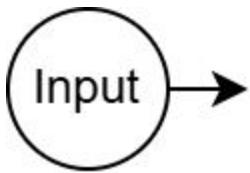


# Construindo um modelo generativo: conceitos

Para facilitar o entendimento de um modelo gerativo, vamos “construir” um modelo gerativo em 4 etapas:

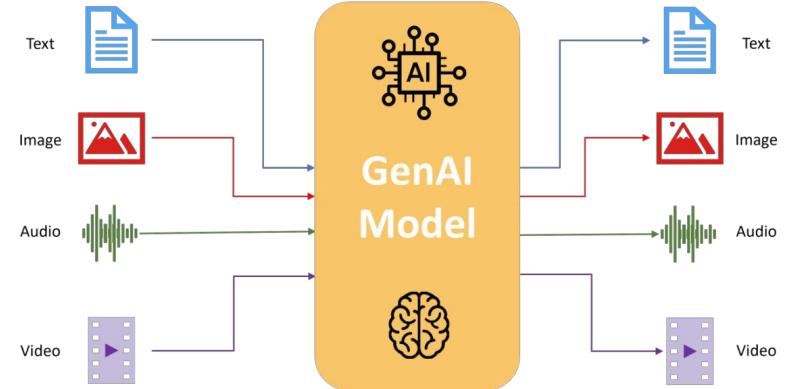
- 1. Inputs e outputs.**
- 2. Tokenização.**
- 3. Embedding.**
- 4. Modelo**

# 1. Inputs e Outputs: Modalidade e Dados.



# 1. Inputs e Outputs: Modalidade

- Geralmente se referimos a modelo gerativo pela sua modalidade, na forma **input-to-output**.
- **Text-to-text**: geração de texto, tradução, correção, Q&A, etc.
- **Text-to-image**: Geração de Imagens por descrição.
- **Image-to-image**: aumento de resolução, pós-processamento, etc.



# 1. Inputs e Outputs: Dados

- Para treinar esses modelos, é necessário uma quantidade massiva de dados da modalidade necessária.
- Os modelos comerciais utilizam quantidades de dados massivas de dados:
  - Web crawling.
  - Dados de usuários em plataformas.
  - Torrent.
  - Dentre outras fontes questionáveis.
- Plataformas com datasets pré processados:
  - **Kaggle** e [Hugging Face](#).

Dataset	# tokens	Proportion within training
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Quantidade de dados utilizada para o treinamento do GPT 3

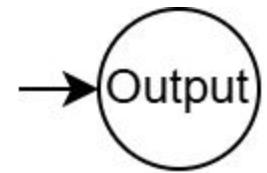
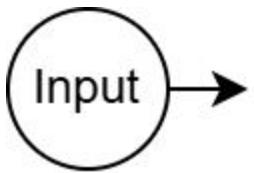


# 1. Inputs e Outputs: Estruturação

- Para treinar algoritmos ML, precisamos de **variáveis de entrada** e uma **variável objetivo**.
  - Não é diferente para modelos generativos.
- Entretanto, dados obtidos da internet são **não-estruturados**.
- Estratégias variam conforme **aprendizado**:
  - Regras pré-estabelecidas separam a variáveis dos rótulos (**auto-supervisionado**).
  - Treinar o modelo parte com rótulos por auto-supervisão com parte com base rotulada por humanos. (**semi-supervisionado**).

Variáveis	Rótulo
$X_{1,1}$	$y_1$
$X_{2,1}$	$y_2$
$X_{3,1}$	$y_3$

# 1. Inputs e Outputs

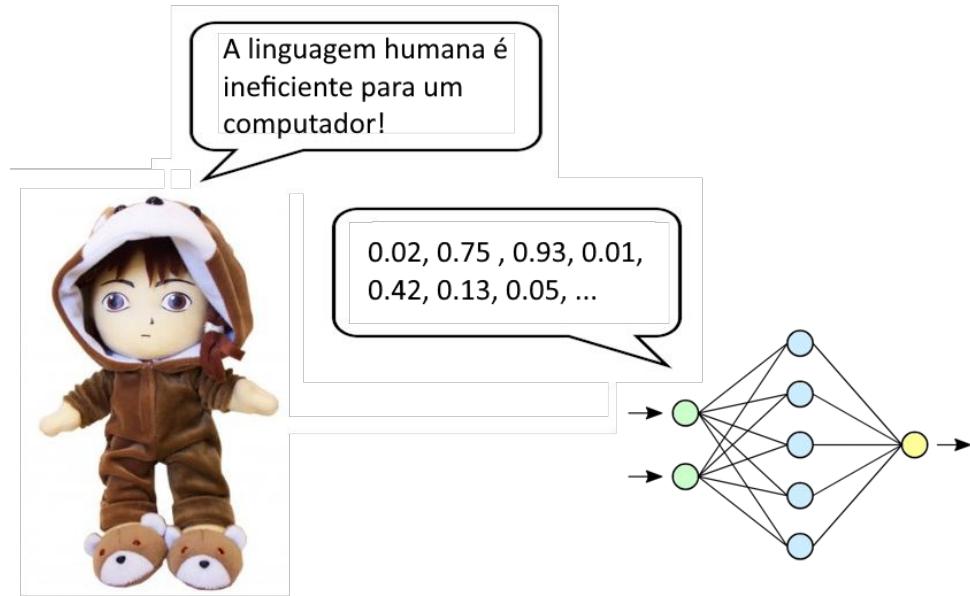


# 1. Inputs e Outputs



# 2. Tokenização: Introdução

- Redes neurais e outros modelos ML lidam somente com valores numéricos.
- Portanto, é necessário converter nossa entrada (texto, imagem, etc) para números:
- Como fazer:
  - a. Fragmentar nosso domínio na menor unidade possível (**token**).
  - b. Criar um mapeamento de tokens para números (**Lexicon**).
- Este processo é denominado **tokenização**.



## 2. Tokenização: Texto

- Como vocês fariam um mapeamento texto para números?

## 2. Tokenização: Texto

- Como vocês fariam um mapeamento texto para números?
  - a. letras -> números.
  - b. palavras -> números.
- Vamos utilizar uma frase como benchmark para nossas alternativas.

O arquiteto não reiniciou o servidor  
porque ele estava com preguiça.

## 2. Tokenização: Texto

- Como vocês fariam um mapeamento texto para números?
  - **letras -> números.**
  - palavras -> números.
- Obtemos 68 tokens
- Agora pense que precisamos compilar todos os livros, artigos e textos da internet para dentro do modelo.
  - Ineficiente.

<sup>15</sup> 1 181721

O arquiteto não reiniciou o servidor  
porque ele estava com preguiça.

[15, 0, 1, 18, 17, 21, 9, 20, 5, 20, 15, 0, 14, 1, 15, 0, 18, 5, 9,  
14, 9, 3, 9, 15, 21, 0, 15, 0, 19, 5, 18, 22, 9, 4, 15, 18, 0, 16,  
15, 18, 17, 21, 5, 0, 5, 12, 5, 0, 5, 19, 20, 1, 22, 1, 0, 3, 15,  
13, 0, 16, 18, 5, 7, 21, 9, 3, 1]

Quantidade de Tokens: 68

## 2. Tokenização: Texto

- Como vocês fariam um mapeamento texto para números?
  - a. letras -> números.
  - b. **palavras -> números.**
- Nesta abordagem podemos utilizar um dicionário como referência.
- Quantidade de tokens caiu mais de 5 vezes.

219195      26887      213002      267358      219195      281596  
O arquiteto não reiniciou o servidor

252197      104165      119840      76912      254276  
porque ele estava com preguiça.

[219195, 26887, 213002, 267358, 219195, 281596,  
252197, 104265, 119840, 76912, 254276]

Quantidade de Tokens: 11

## 2. Tokenização: Texto

- Como vocês fariam um mapeamento texto para números?
  - a. letras -> números.
  - b. palavras -> números.
- Agora, pense na perspectiva de um modelo de linguagem.
  - a. Precisamos entender o contexto.
  - b. Preguiçoso, arquitetura, reiniciar, são palavras que poderiam surgir em uma conversa continuando essa frase.
  - c. **É possível tokenizar palavras por significado?**

O arquiteto não reiniciou o servidor porque ele estava com preguiça.

## 2. Tokenização: Texto

- Como vocês fariam um mapeamento texto para números?
  - a. letras -> números.
  - b. palavras -> números.
  - c. **subpalavras -> números.**
- A palavra preguiça compartilha o mesmo morfema radical com preguiçoso, por exemplo.
- Essa interpretação pode gerar mais tokens, mas é mais facilmente compreendida pelo modelo.

Tokens	Characters
18	68
0 arquiteto não reiniciou o servidor porque ele estava com preguiça.	

Tokenização em subpalavras por Byte-Pair Encoding (BPE) no chat GPT4

## 2. Tokenização: Imagens

- Como aplicamos tokens em imagens?

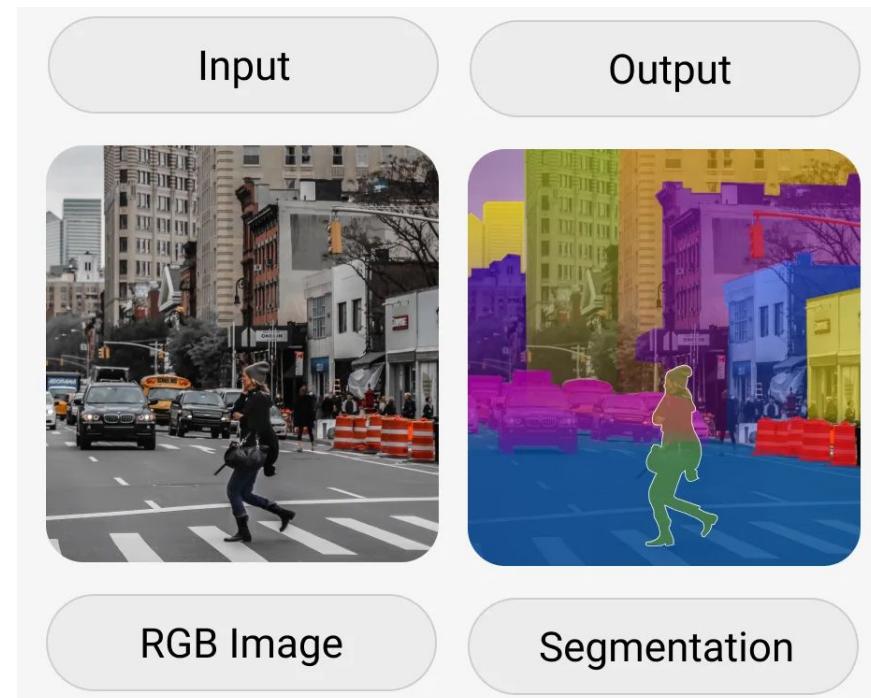
## 2. Tokenização: Imagens

- Como aplicamos tokens em imagens?
- Podemos realizar tokenização de forma semelhante a texto:
  - Divisão da imagem regiões (**patches**)
- Entretanto, tal divisão não é tão eficiente para imagens:
  - Subimagens não preservam significado da mesma forma que subpalavras.



## 2. Tokenização: Imagens

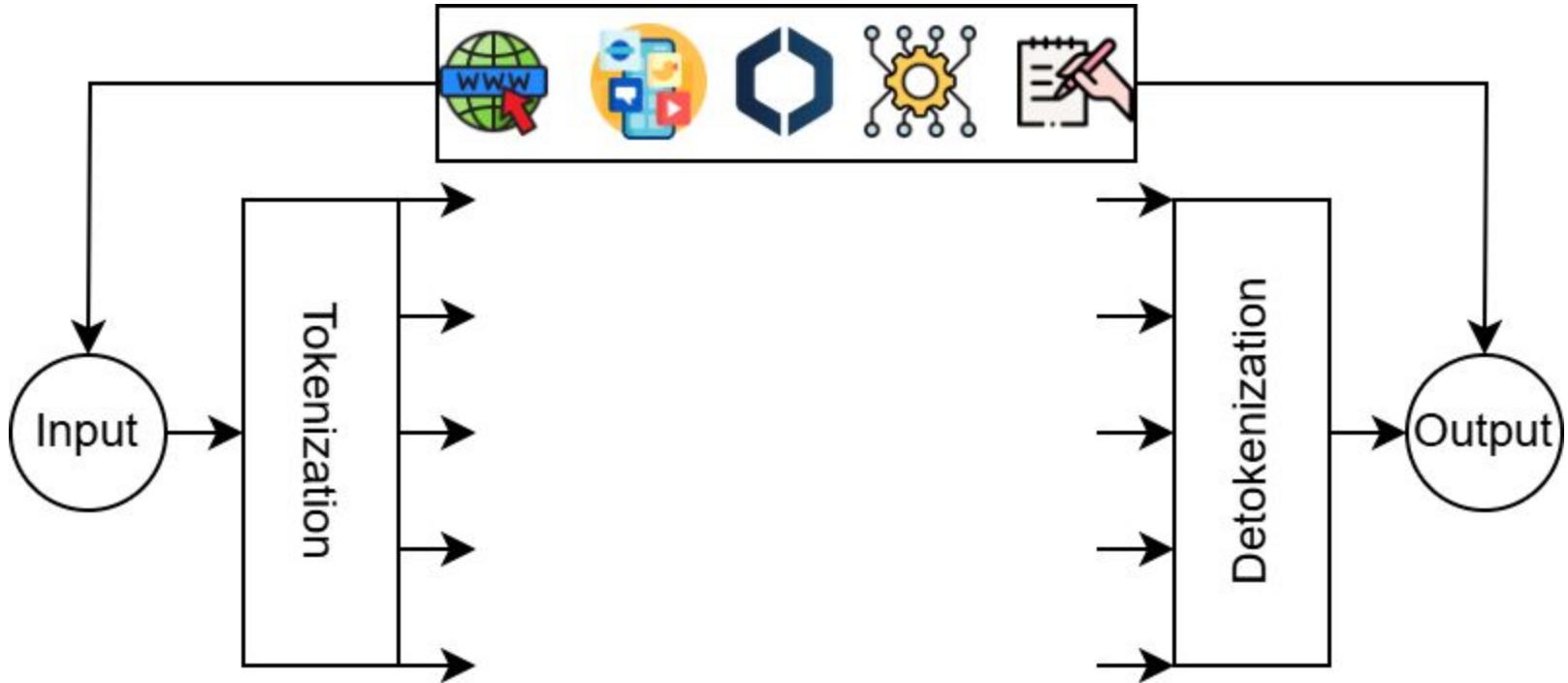
- Como aplicamos tokens em imagens?
- Podemos realizar tokenização de forma semelhante a texto:
  - Divisão da imagem regiões (**patches**)
- Entretanto, tal divisão não é tão eficiente para imagens:
  - Subimagens não preservam significado da mesma forma que subpalavras.
- A tokenização das imagens é feita por meio de **convoluções**:
  - Os patches não são uma tokenização dos pixels, mas sim dos padrões.



## 2. Tokenização



## 2. Tokenização



# ■ 3. Embedding: Introdução

- Se coloquem na perspectiva de uma gen AI por um momento:

# 3. Embedding: Introdução

- Se coloquem na perspectiva de uma gen AI por um momento:



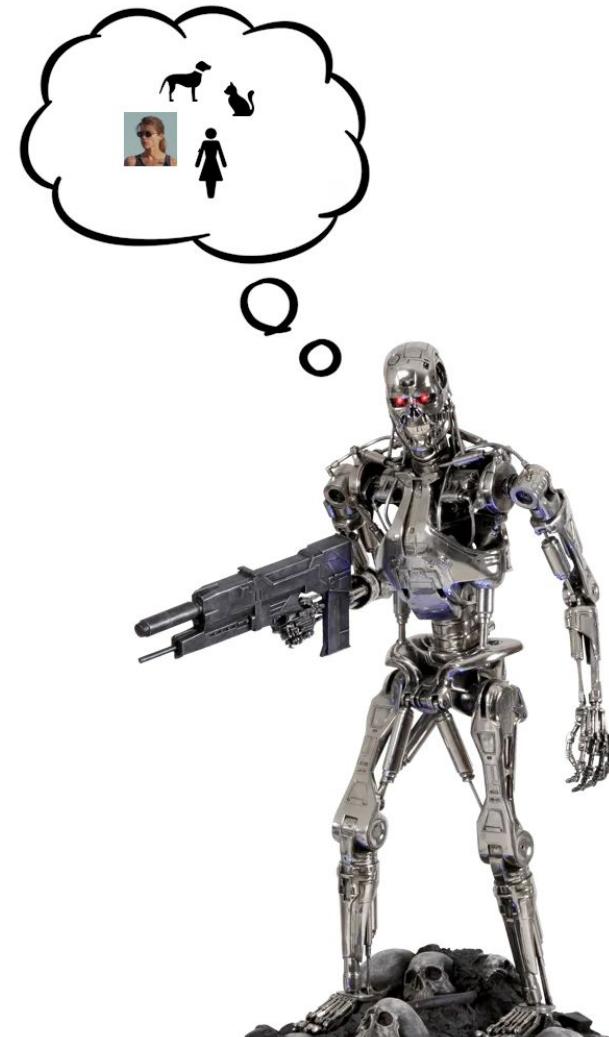
# 3. Embedding: Introdução

- Se coloquem na perspectiva de uma gen AI por um momento:
  - Se todo **conceito** é atribuído um **número** (token), qual lógica utilizamos para atribuir números a conceitos?



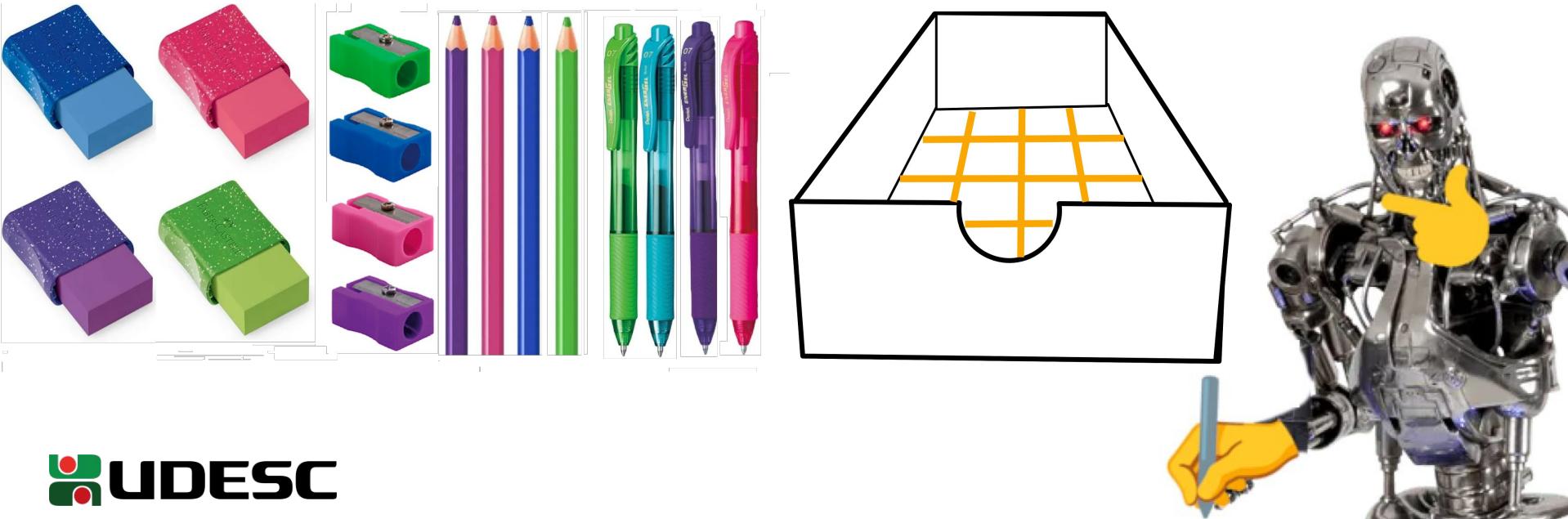
# 3. Embedding: Introdução

- Se coloquem na perspectiva de uma gen AI por um momento:
  - Se todo **conceito** é atribuído um **número** (token), qual lógica utilizamos para atribuir números conceitos?
- Pense em conceitos como: gato, cachorro, mulher, Sarah Connor, etc.
  - Como organizar?

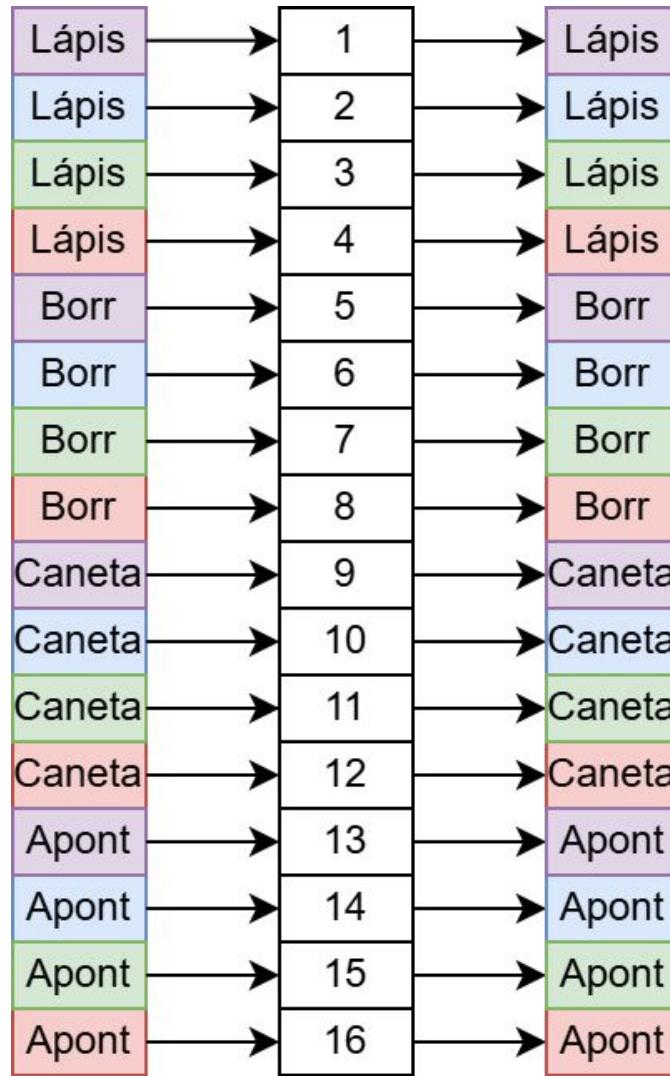


# 3. Embedding: Exemplo

- Vamos fazer uma pequena dinâmica:
  - Imagine que você tem 16 itens escolares de tipos e cores diferentes que precisam ser organizados em uma gaveta.
  - Como fazer isso de modo que:
    - Sempre saiba onde cada item está.
    - Gere o mínimo de notas possível.
    - Itens estejam organizados por semelhança



# 3. Embedding: Exemplo

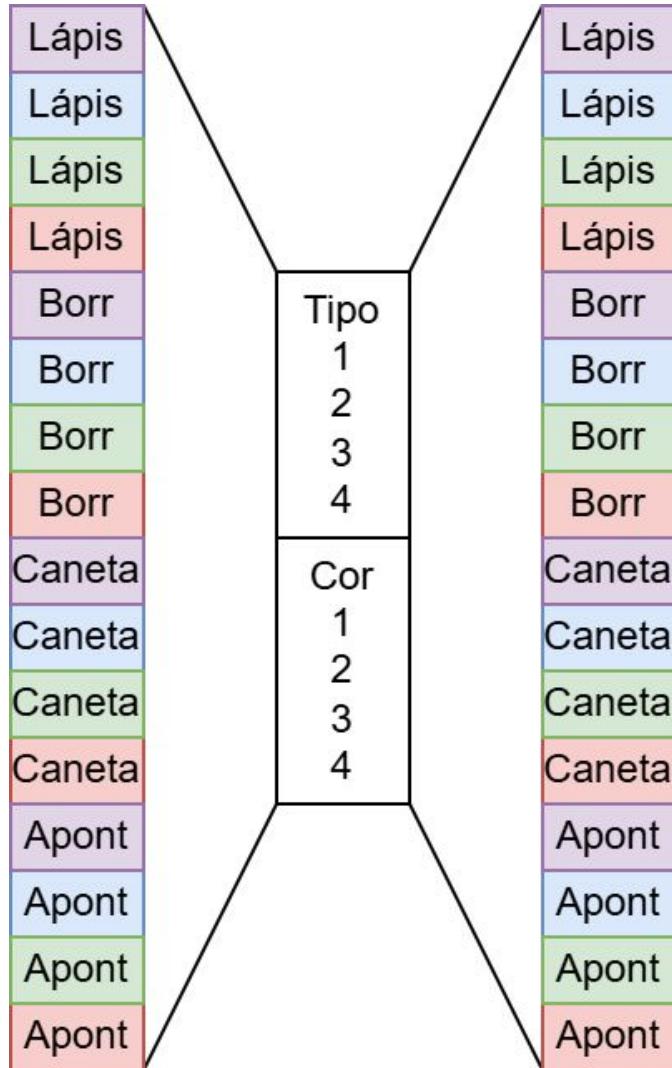


- Forma 1: Mapeamento direto.
  - Tradução Rápida.
  - Não existe relação semântica entre a posição e o item.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



# 3. Embedding: Exemplo



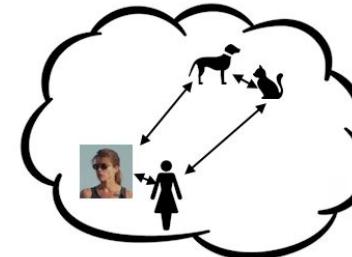
- Forma 2: Agrupamento por conceito.
  - **Maior processamento, mas Menor dimensionalidade:** 16 posições são representadas em uma combinação lógica de 8 informações (4 tipos e 4 cores).
  - **Semântica:** Valores mais próximos são semelhantes.

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	4,4
4,1	4,2	4,3	4,4



# 3. Embedding: Introdução

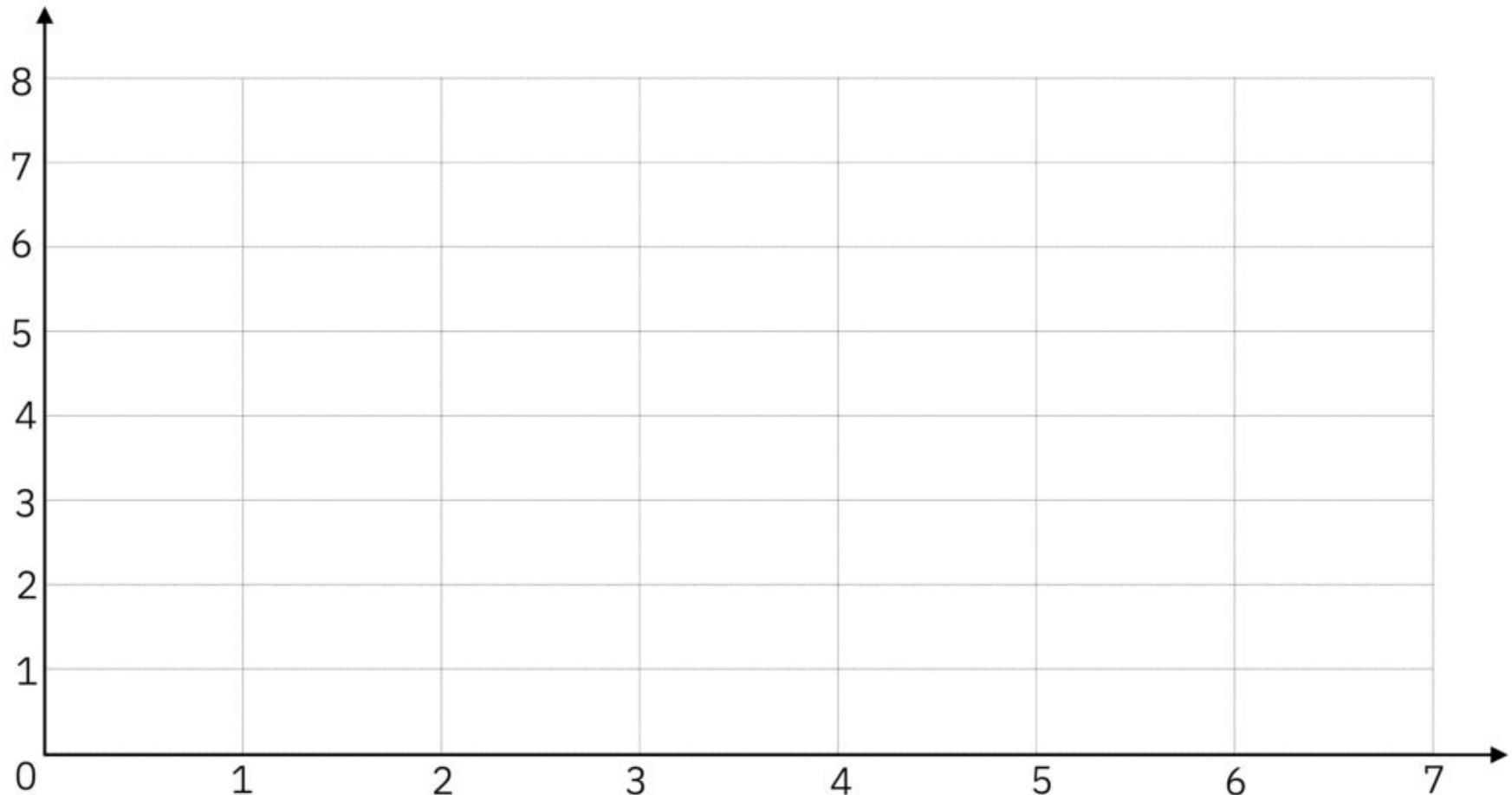
- **Podemos mapear valores por similaridade!**
  - Mais eficiência.
  - Proximidade de conceitos.
- Nesse sentido, duas coisas são importantes:
  - Semântica.
  - Contexto.
- O **embedding** é o mapeamento por similaridade dos tokens para um **espaço vetorial de menor dimensão**, denominado **espaço latente**.



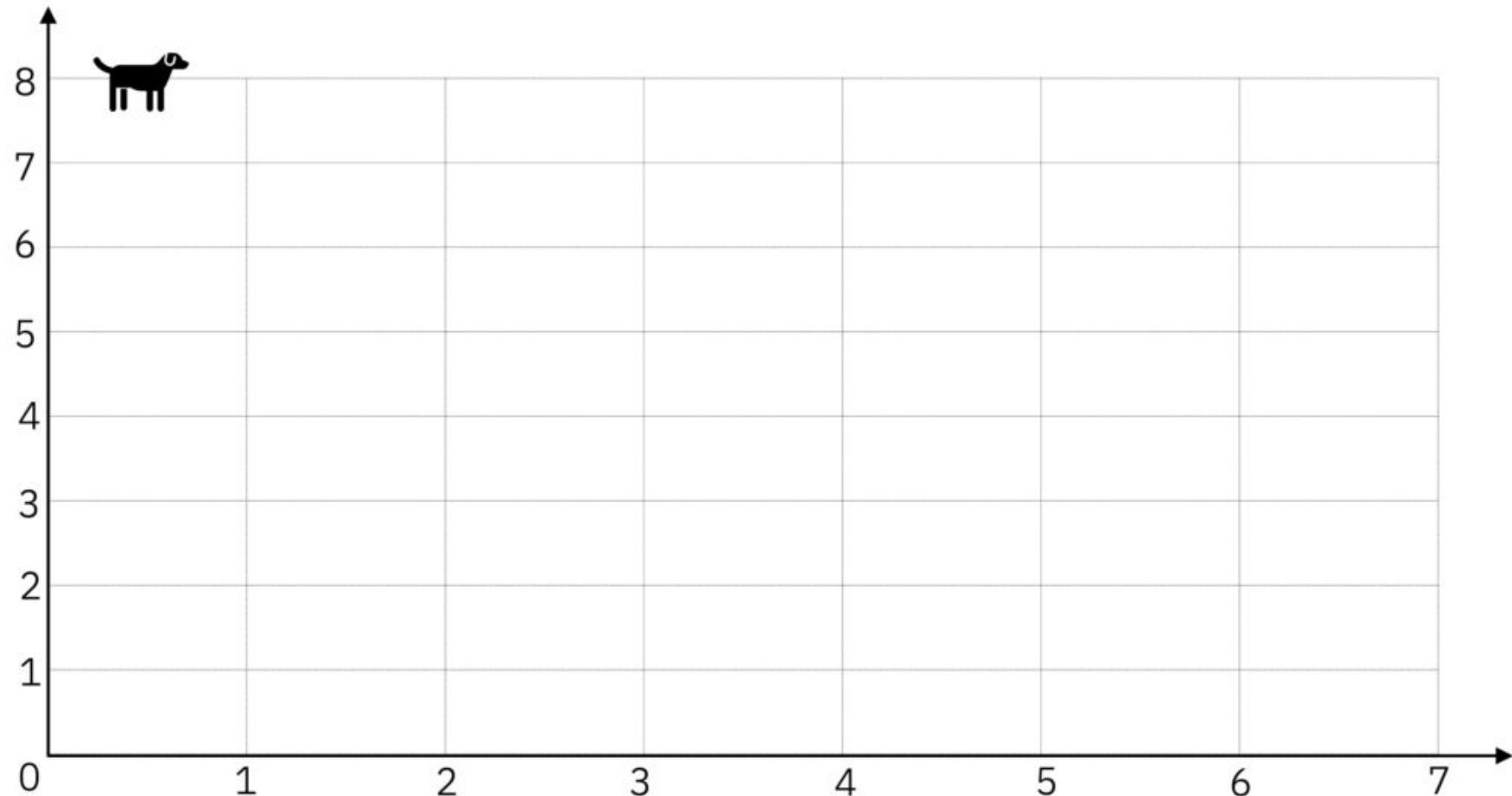
Q<sub>O</sub>



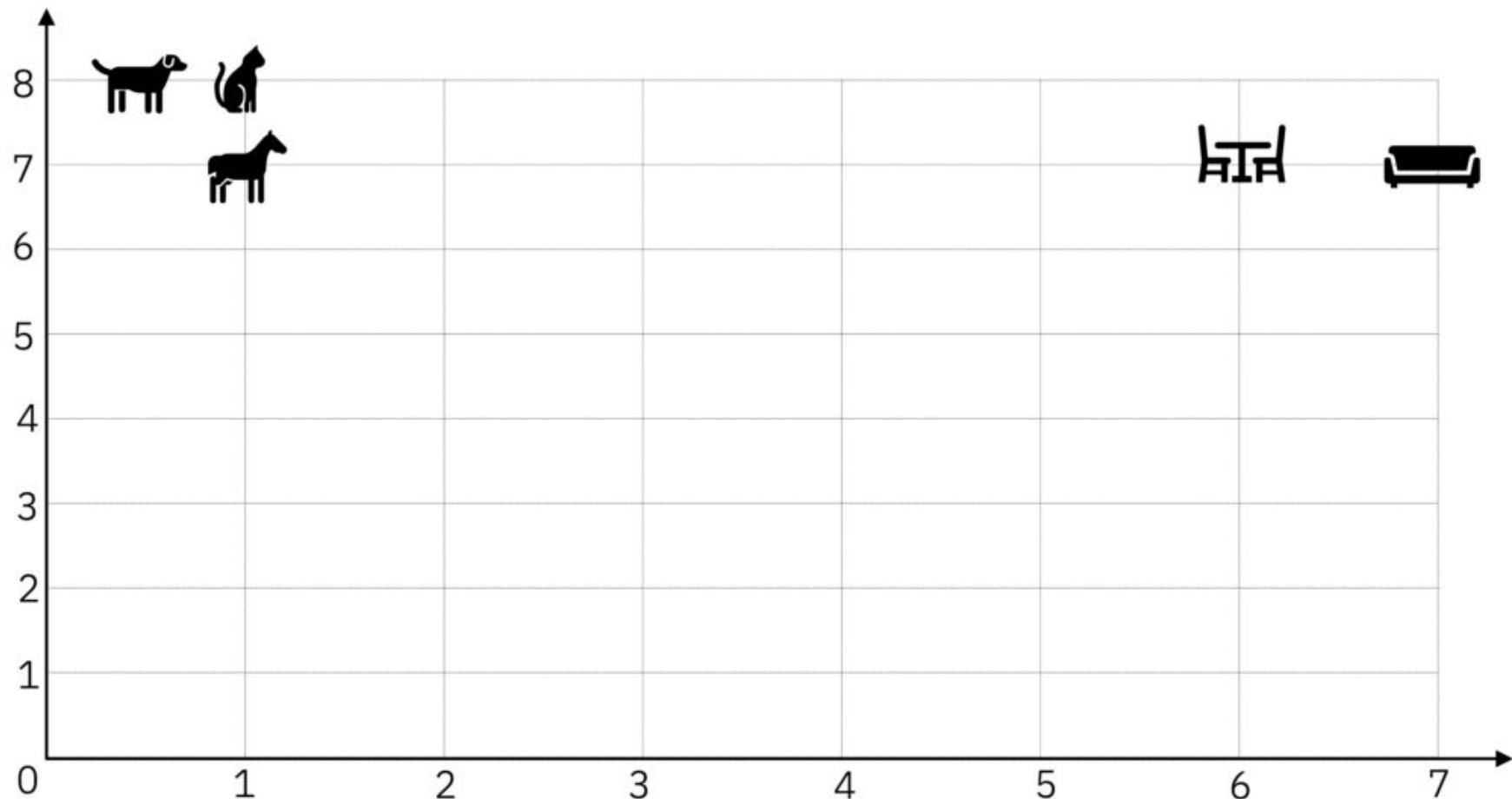
# 3. Embedding: Exemplo com duas dimensões



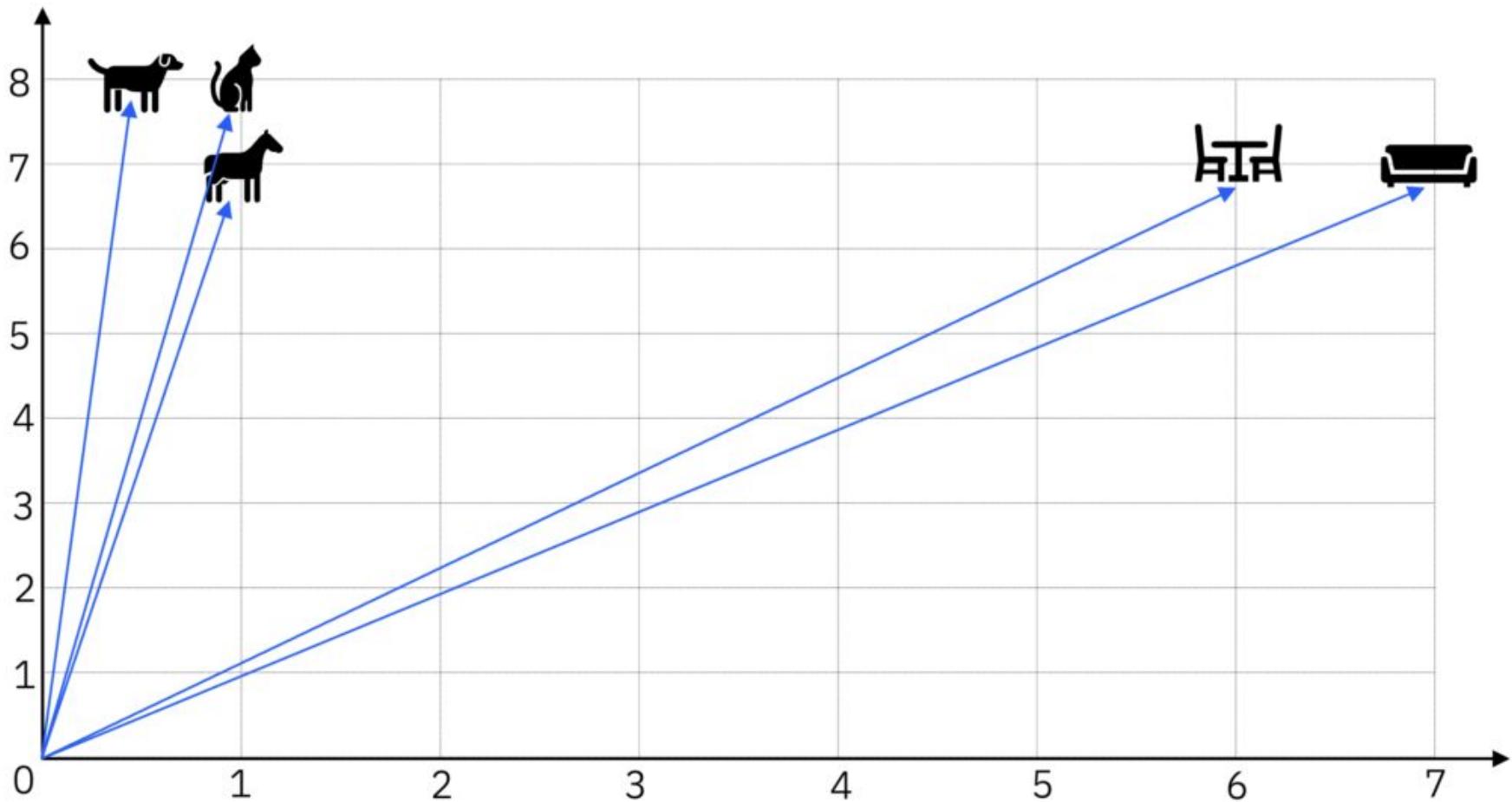
# 3. Embedding: Exemplo com duas dimensões



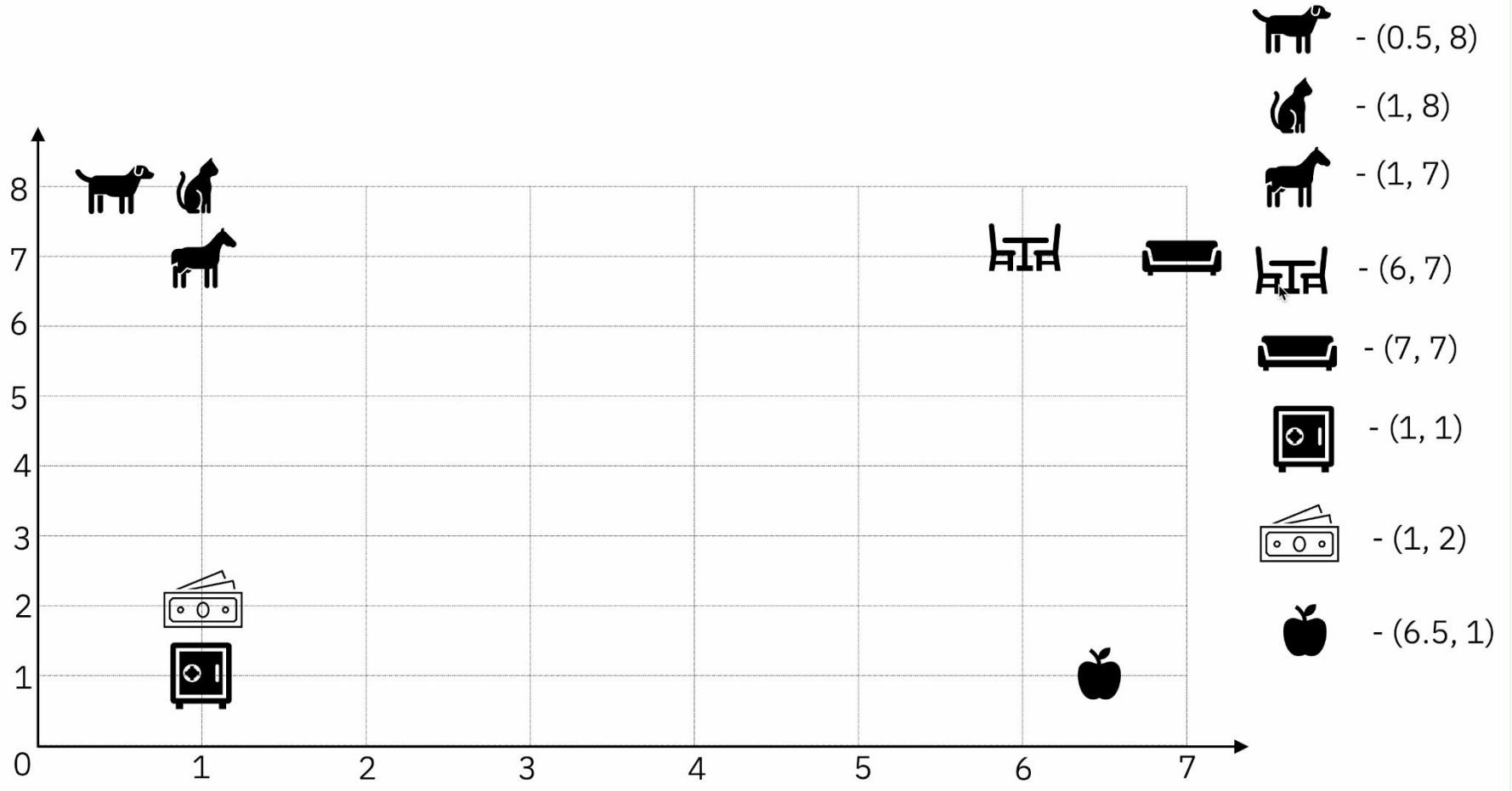
### 3. Embedding: Exemplo com duas dimensões



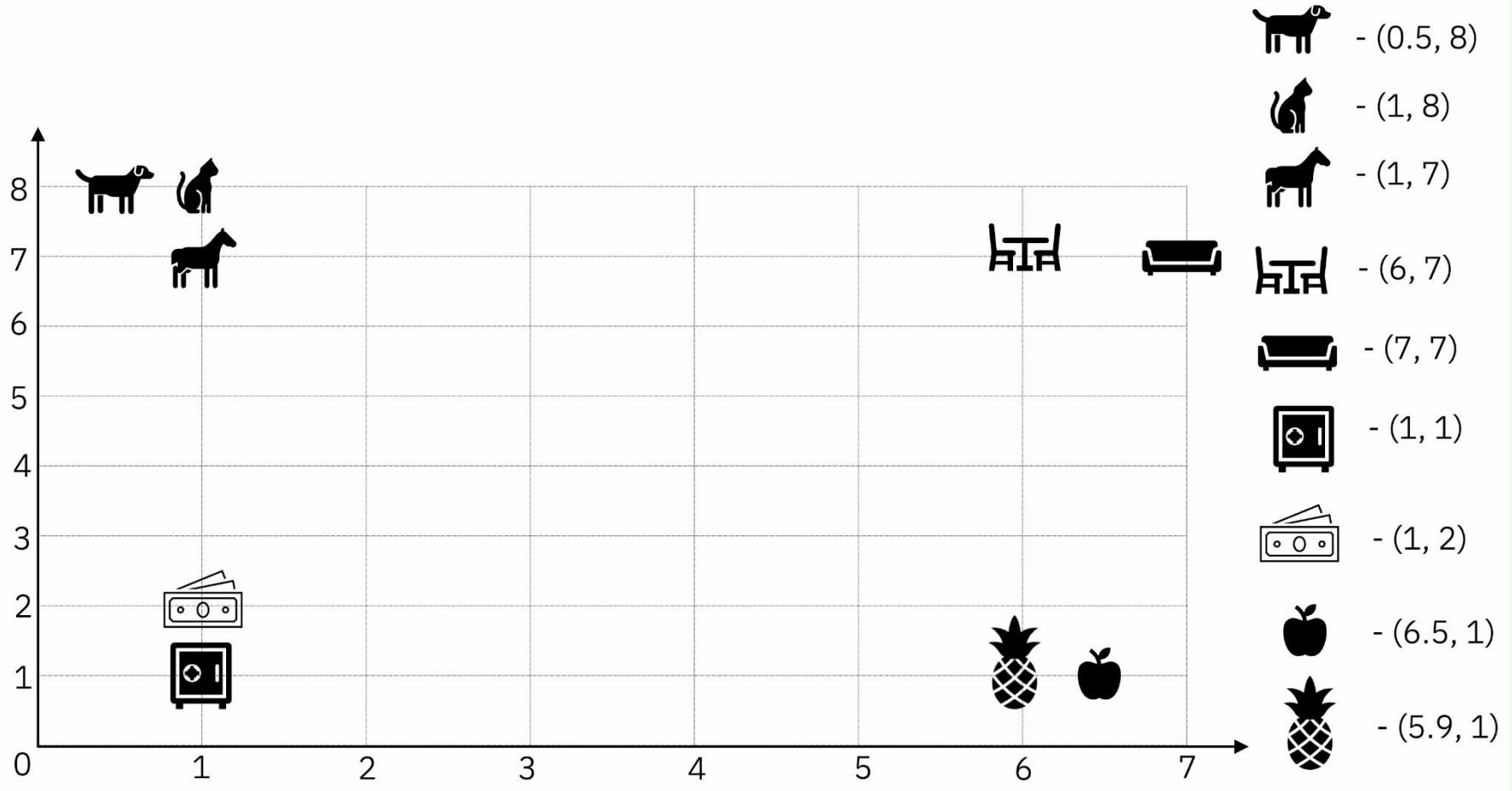
# 3. Embedding: Exemplo com duas dimensões



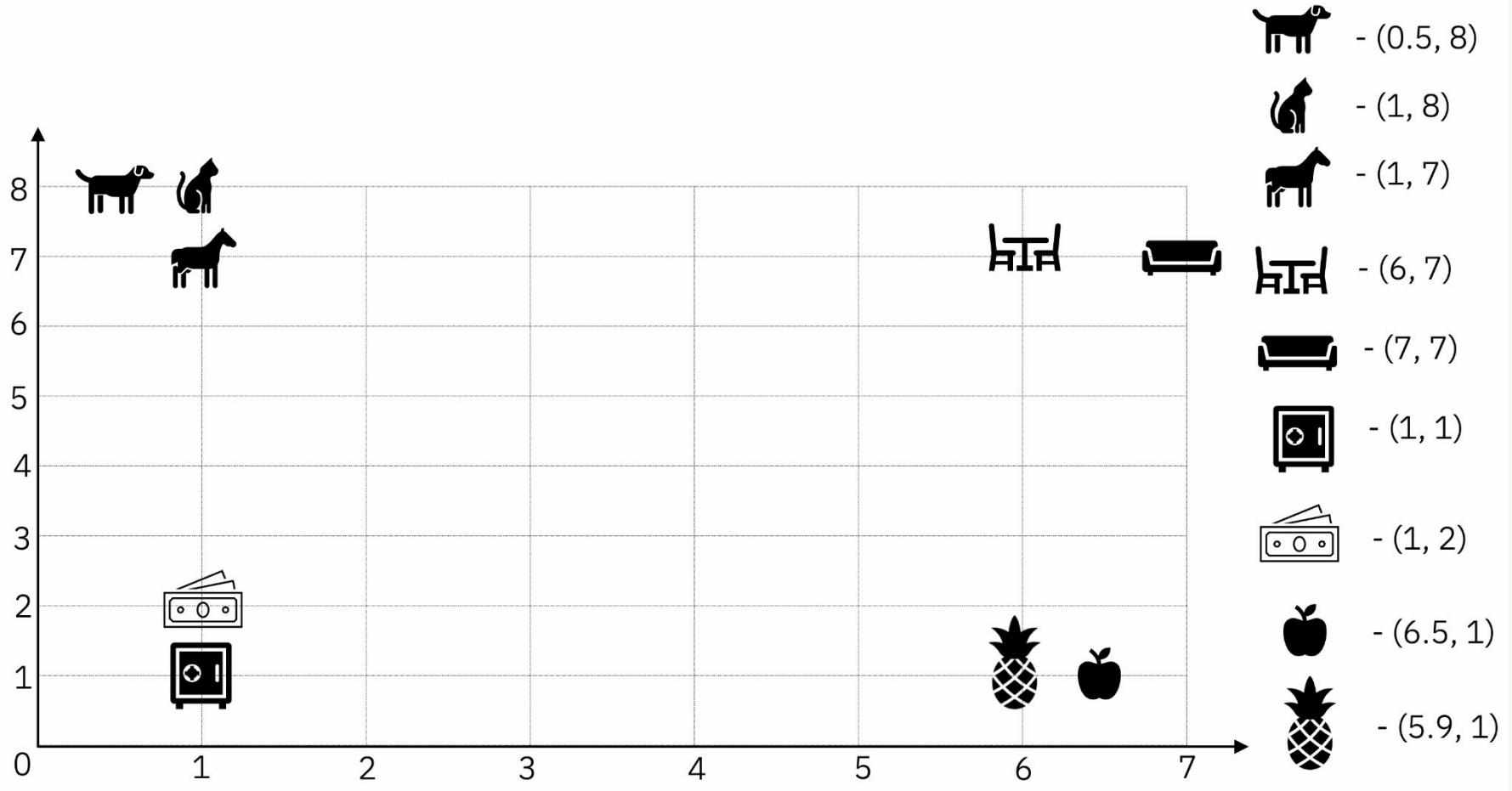
# 3. Embedding: Exemplo com duas dimensões



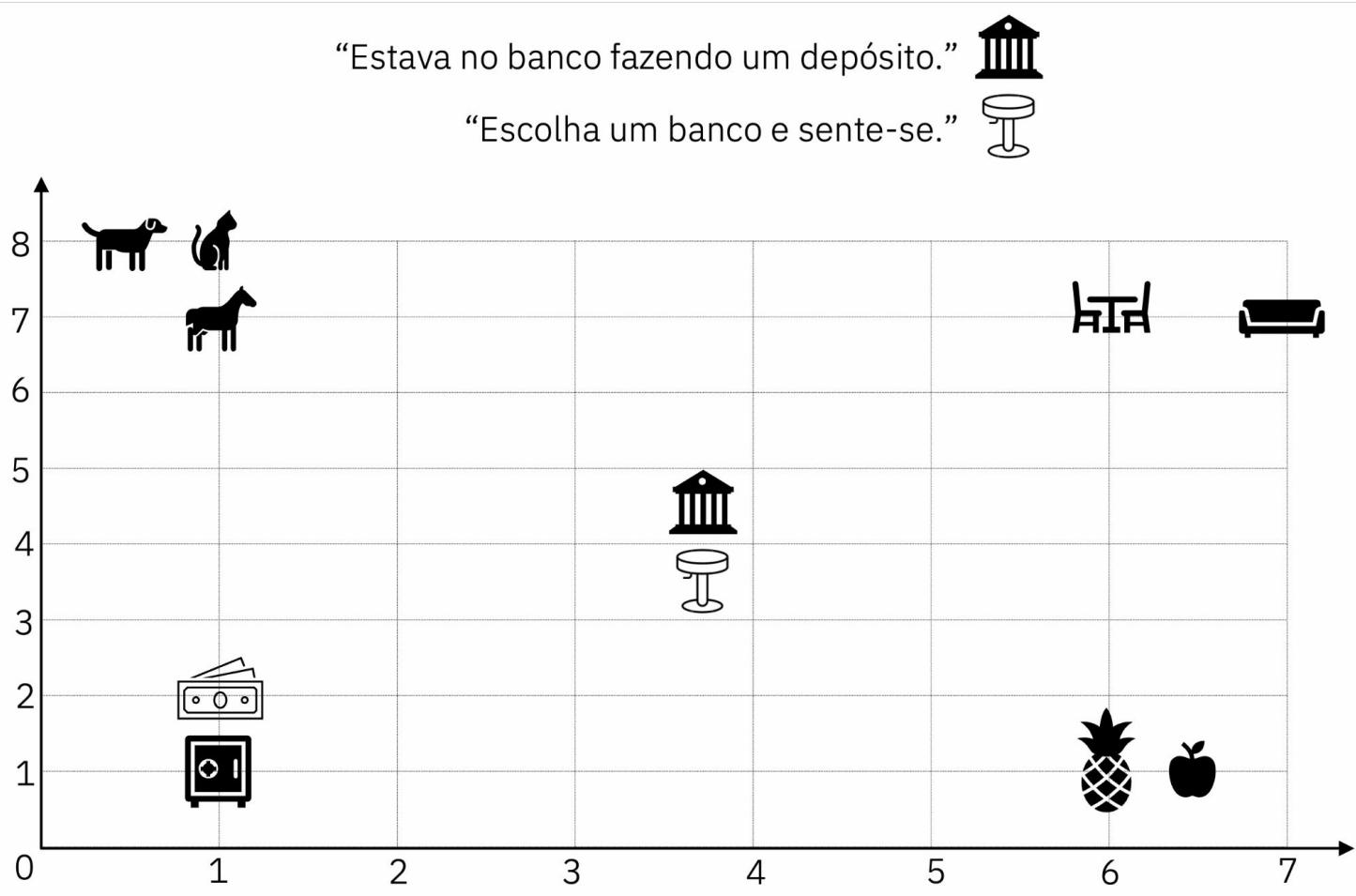
# 3. Embedding: Exemplo com duas dimensões



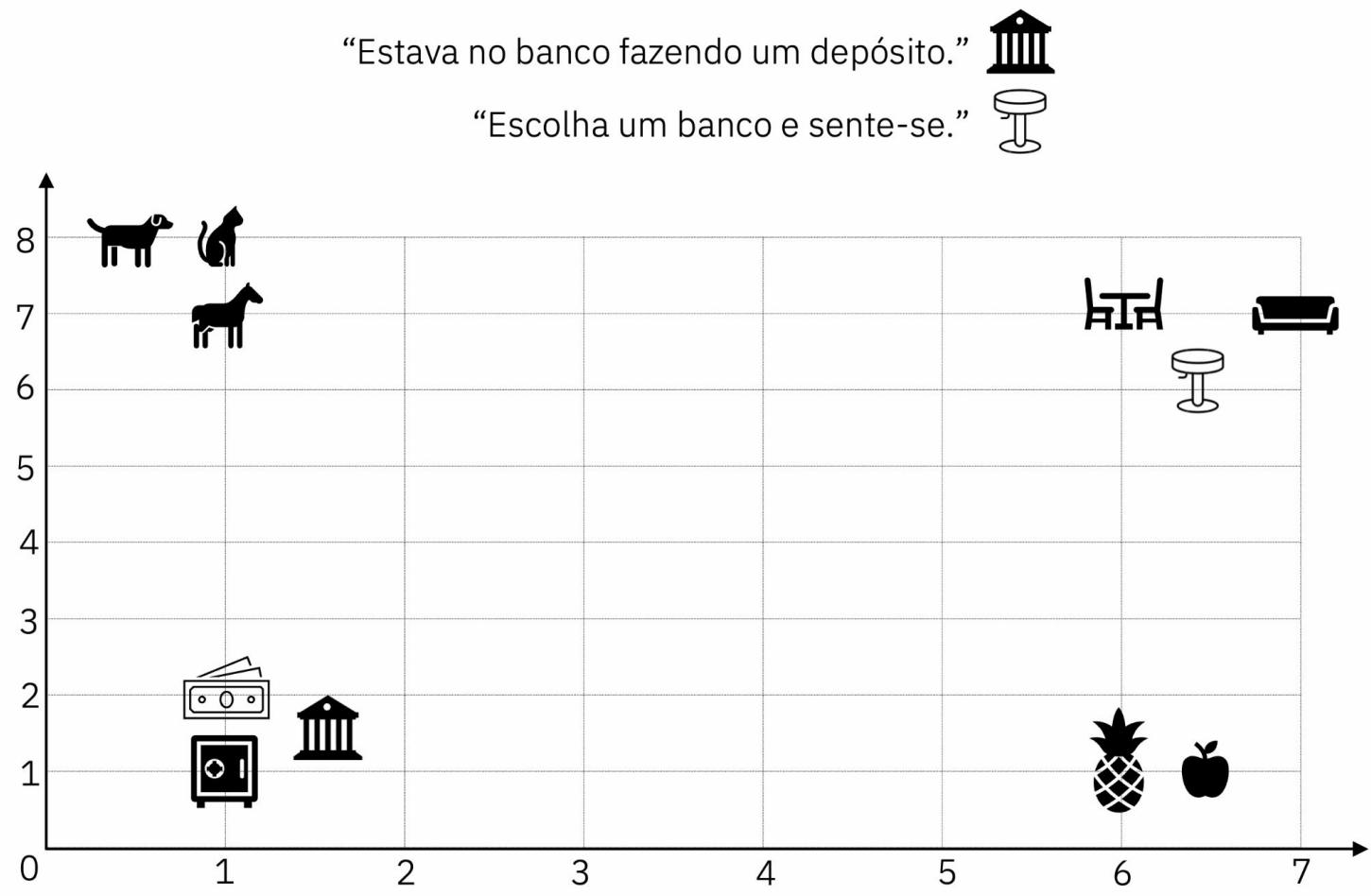
# 3. Embedding: Exemplo com duas dimensões



# 3. Embedding: Exemplo com duas dimensões



# 3. Embedding: Exemplo com duas dimensões



# 3. Embedding: Na prática

- Embeddings reais possuem muito mais dimensões.
- O espaço latente é um **hiperplano**.



[-0.06113929, -0.0012407, 0.06087311, 0.01699911, 0.05108206, ..., 0.03732946, -0.00689885]



[-0.01101368, -0.04874269, -0.05087062, -0.02283244, 0.01541347, ..., 0.06616838, 0.0045159]



[-0.05816573, -0.03017926, 0.05343566, -0.06409686, 0.0160787, ..., -0.0134629, -0.00547542]



[0.04290543, 0.04314668, 0.06709401, -0.02074, -0.0637757, ..., -0.01543431, -0.03469143]

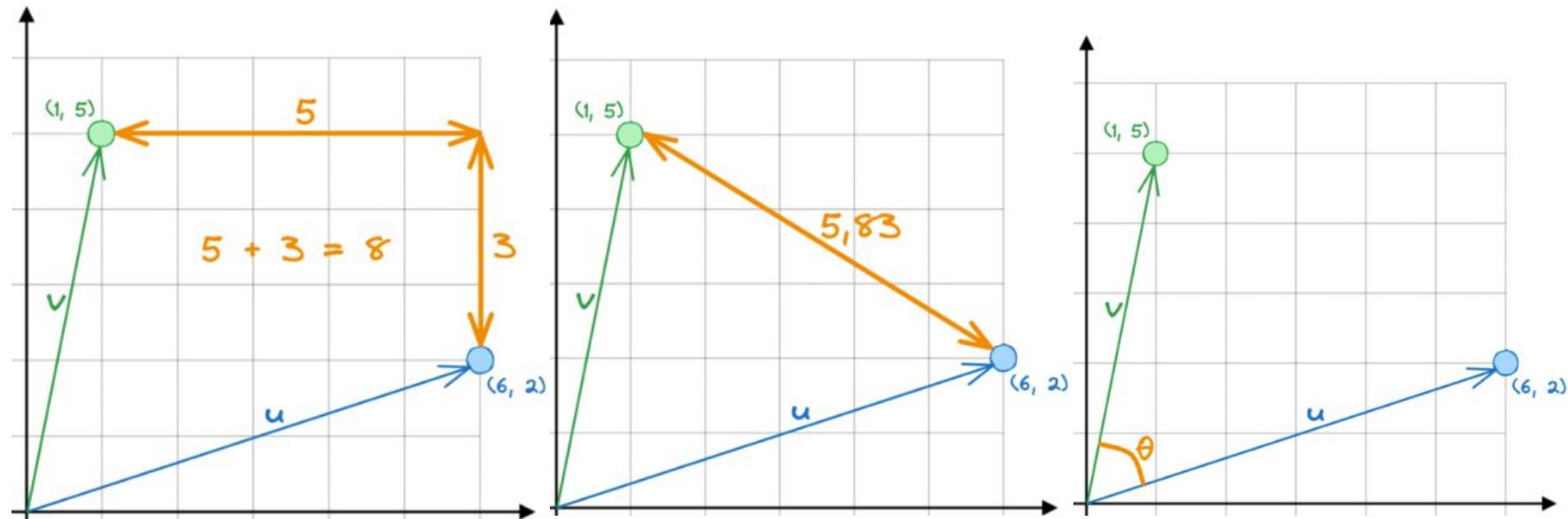


[0.02085212, -0.04604341, -0.0511762, -0.05042295, -0.03493, 0.047325, ..., -0.06708, 0.01174]

512 ~ 4096 dimensões

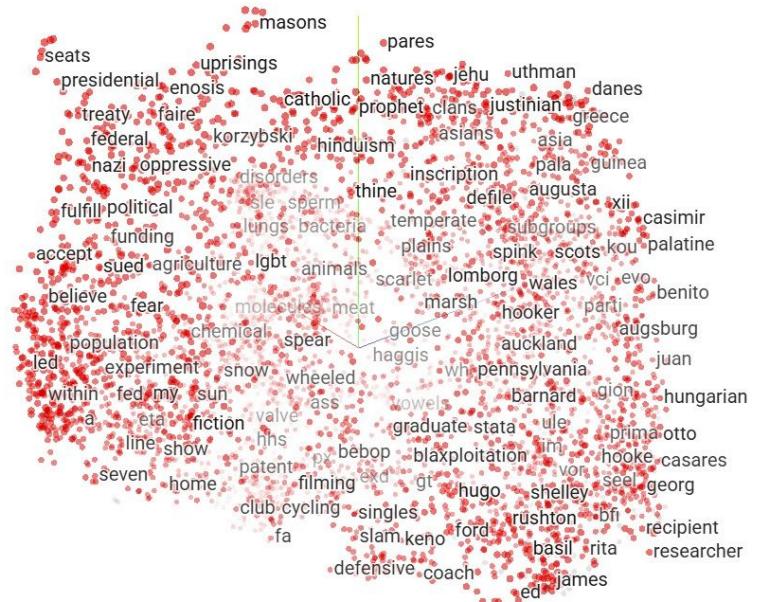
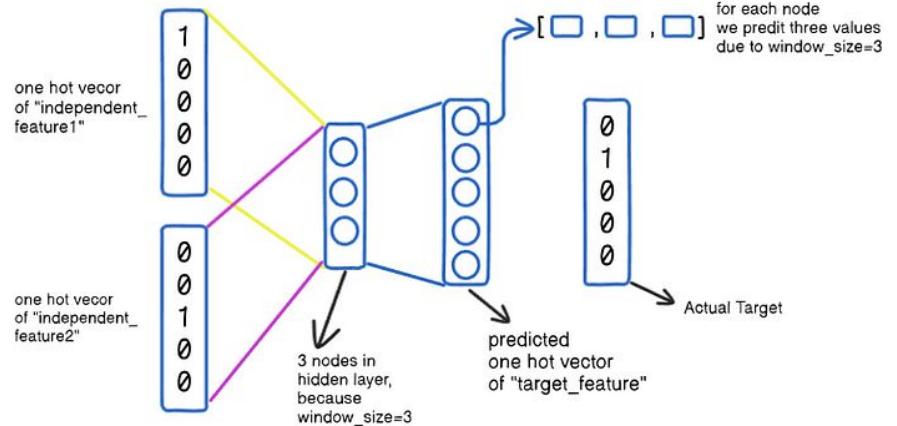
# 3. Embedding: Similaridade

- Medidas de similaridade (distância):
  - Distância do Táxi (Norma L1)
  - Distância euclidiana (Norma L2)
  - **Distância do cosseno.**
    - Menos processamento em alta dimensionalidade.
    - Mais utilizada.



# 3. Embeddings

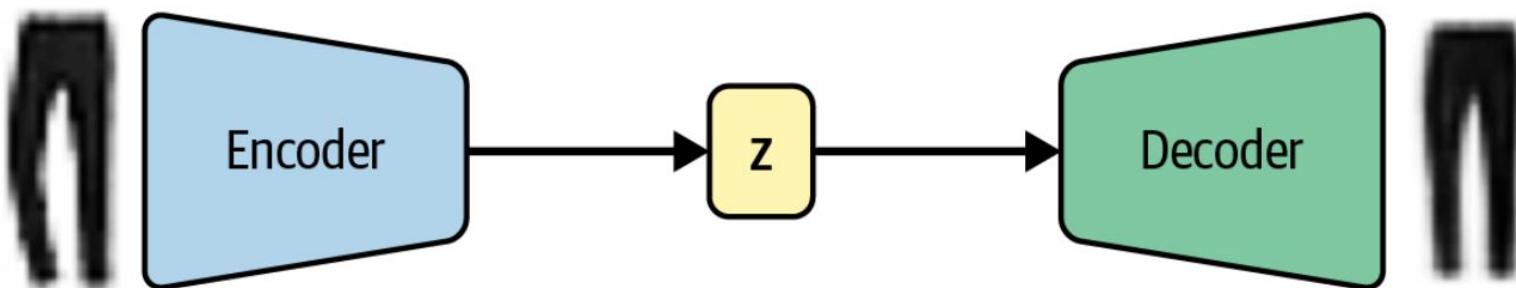
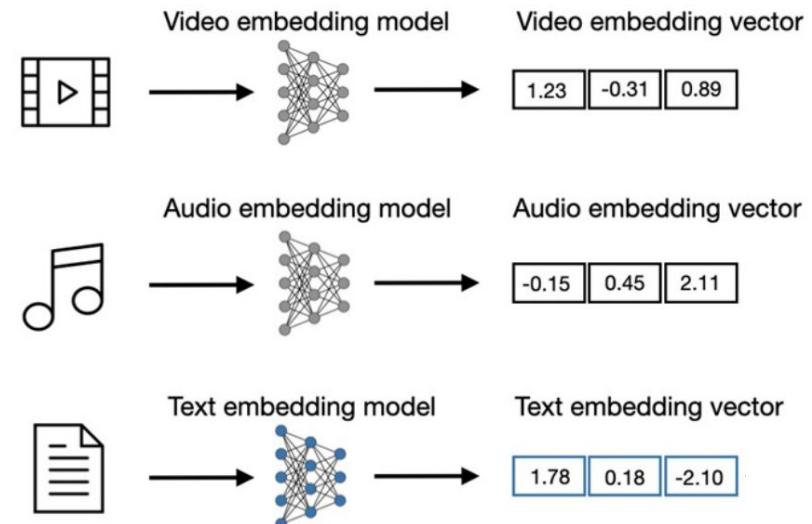
- Os primeiros embeddings não consideravam a semântica, eram apenas um mapeamento dos tokens para vetores genéricos.
- **Embeddings modernos são Redes Neurais Artificiais:**
  - Podem ser treinados separadamente ou junto com o modelo como camadas de embedding.
- **Embeddings estáticos.**
  - Conceitos que ocorrem próximos mais vezes == mais próximos.
  - Estáticos e modulares.
  - Ex: Word2Vec, GloVe, CLIP.



Word2Vec é um famoso embedding gerado utilizando uma Rede Neural Artificial

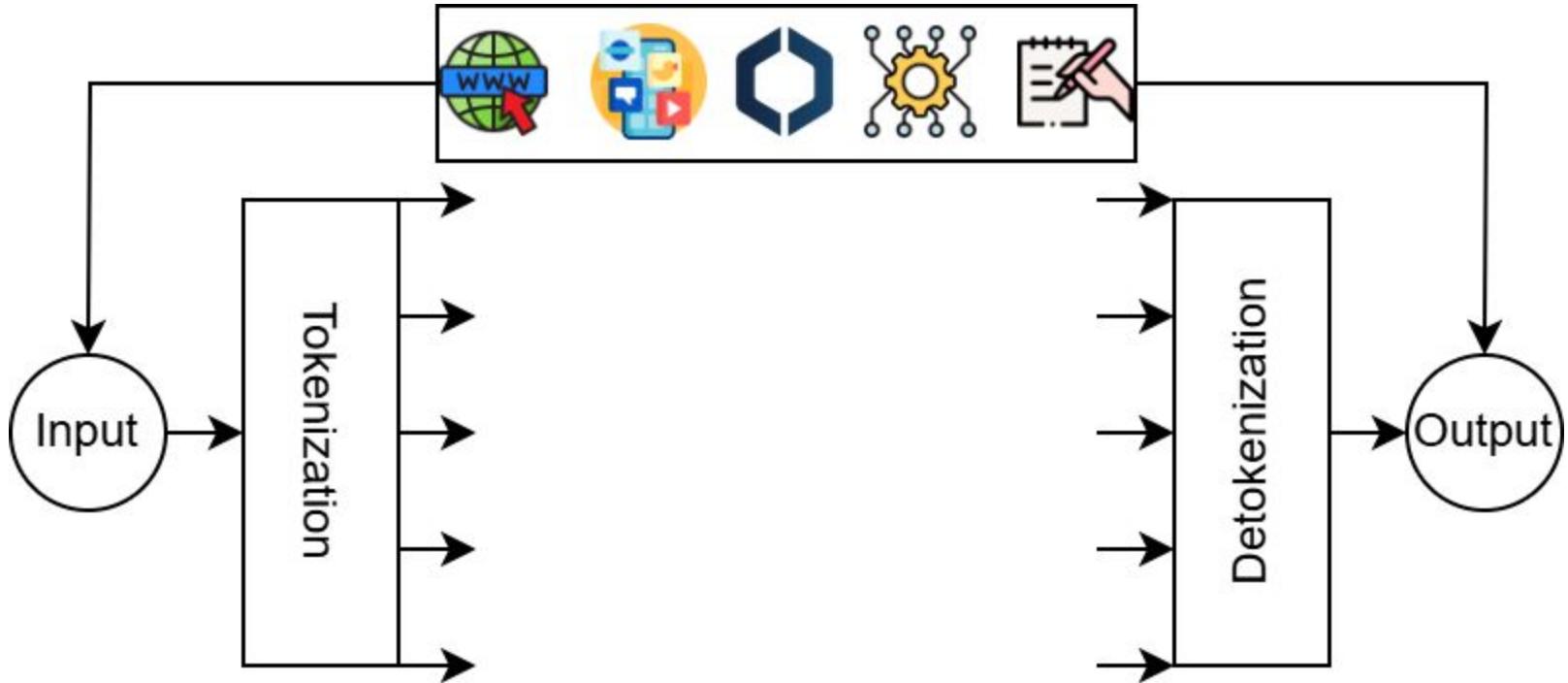
# 3. Como construir embeddings

- Embeddings são construídos com ANNs **Encoder/Decoder**:
- O modelo reconstrói as entradas passando por uma **camada de gargalo**.
- O valor de uma entrada na camada de gargalo se torna seu embedding.
- Encoder: Compressão→gargalo
- Decoder: Gargalo→descompressão

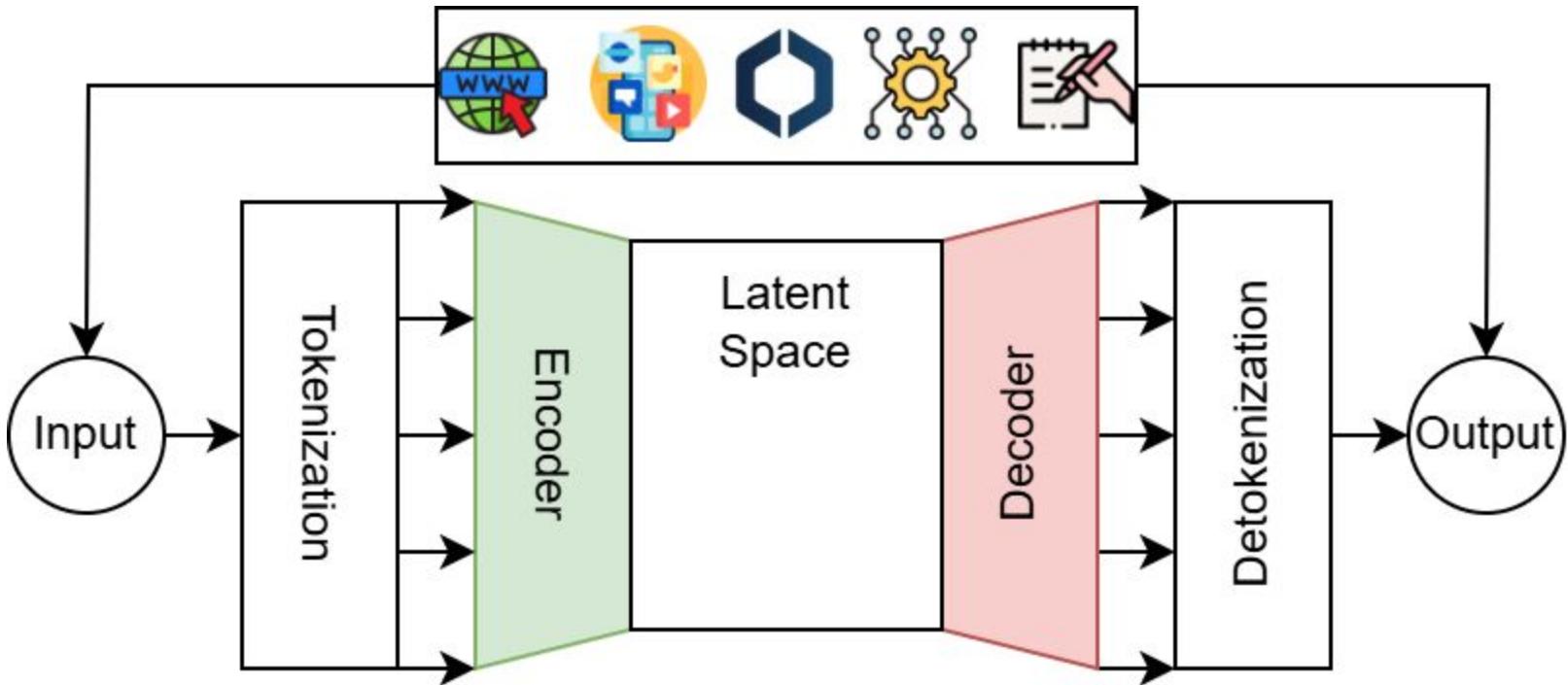


Representação de um autoencoder

# 3. Embedding

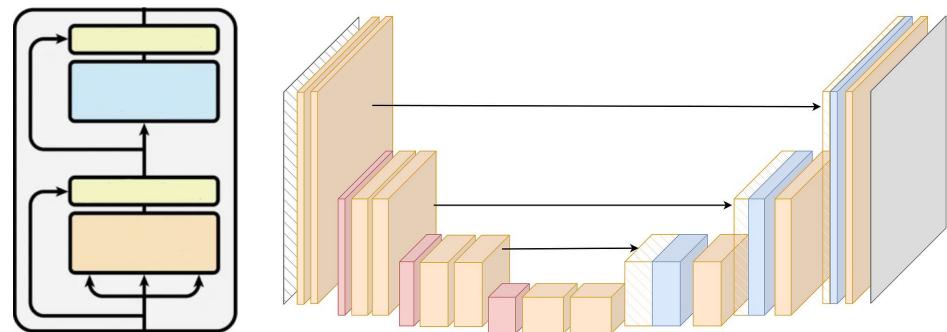
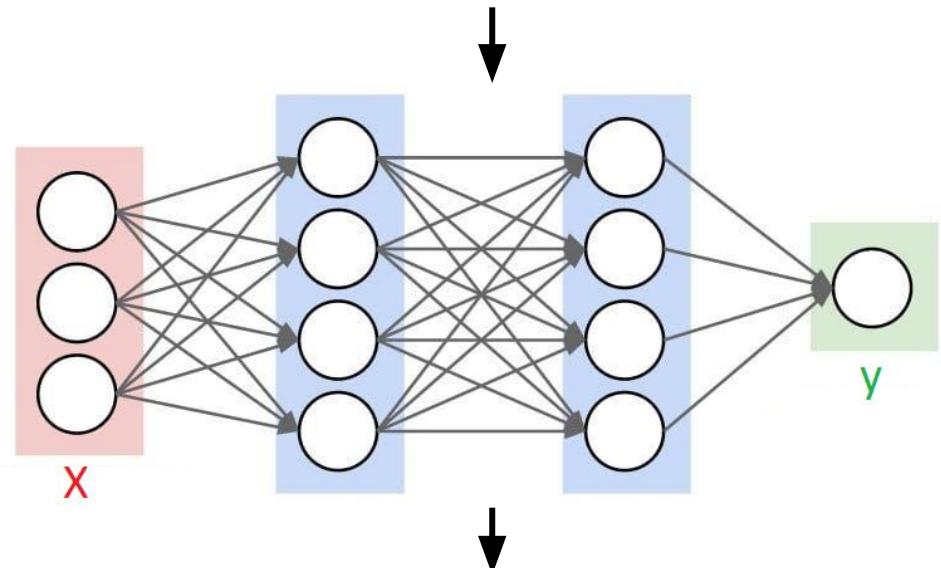


# 3. Embedding



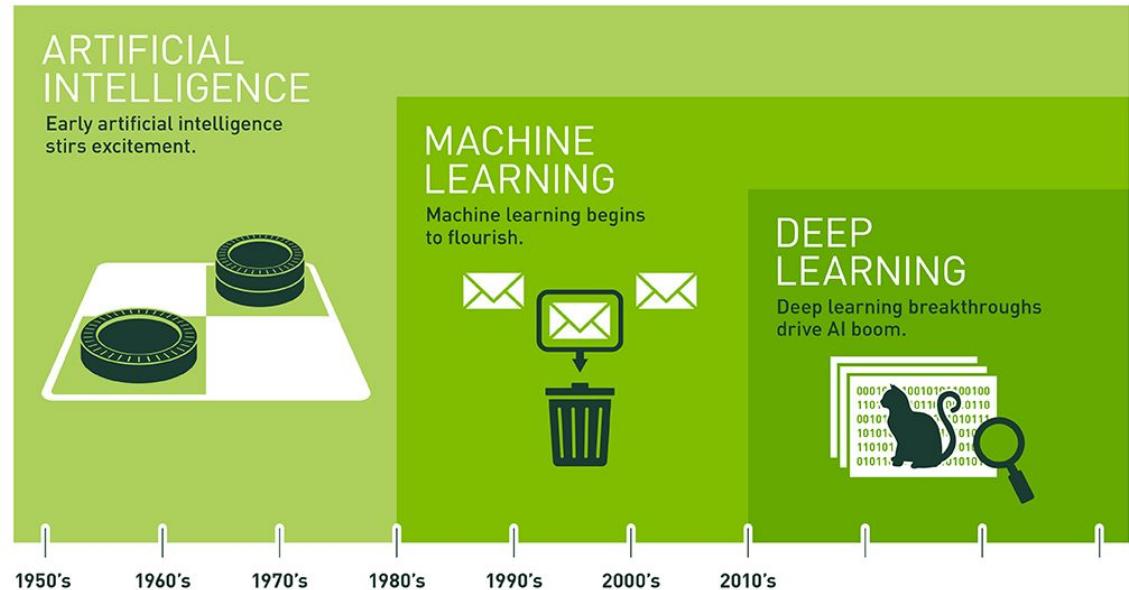
# 4. Arquitetura do Modelo: Introdução

- ANNs organizadas em **camadas interligadas** compõem a arquitetura denominada **Multilayer Perceptron (MLP)**.
- **A MLP é a base de modelos generativos.**
- A forma como as camadas de um MLP são organizadas são referidos como **arquiteturas**.



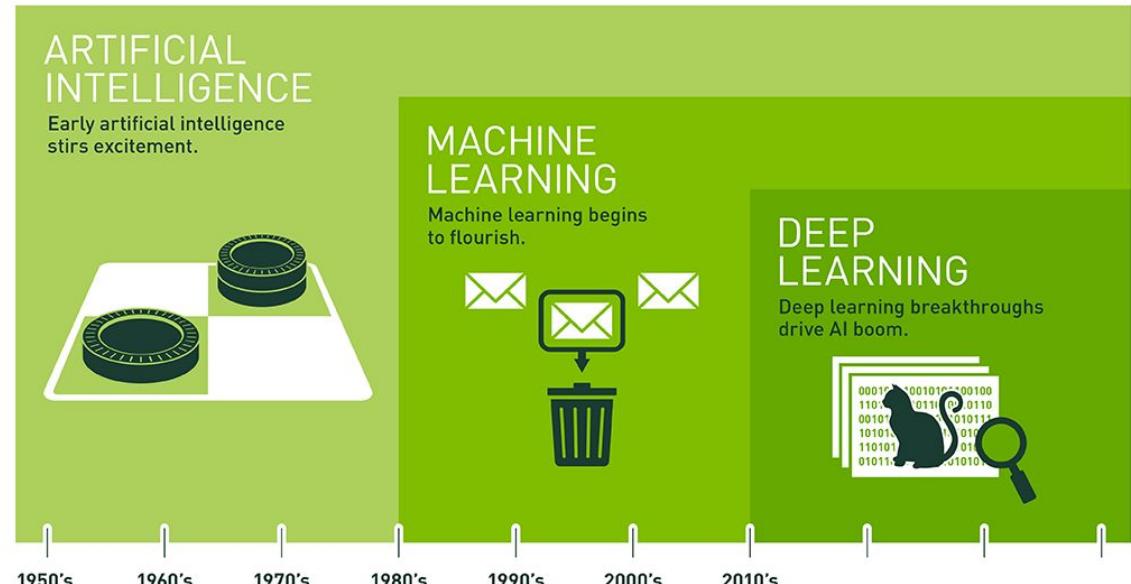
# 4. Modelo: Redes Neurais Profundas

- Quase todos os modelos gerativos têm uma rede neural profunda como modelo fundacional.
  - **Várias camadas**
  - **Alto nível.**
- Área se desenvolveu em 2010. **Por que?**

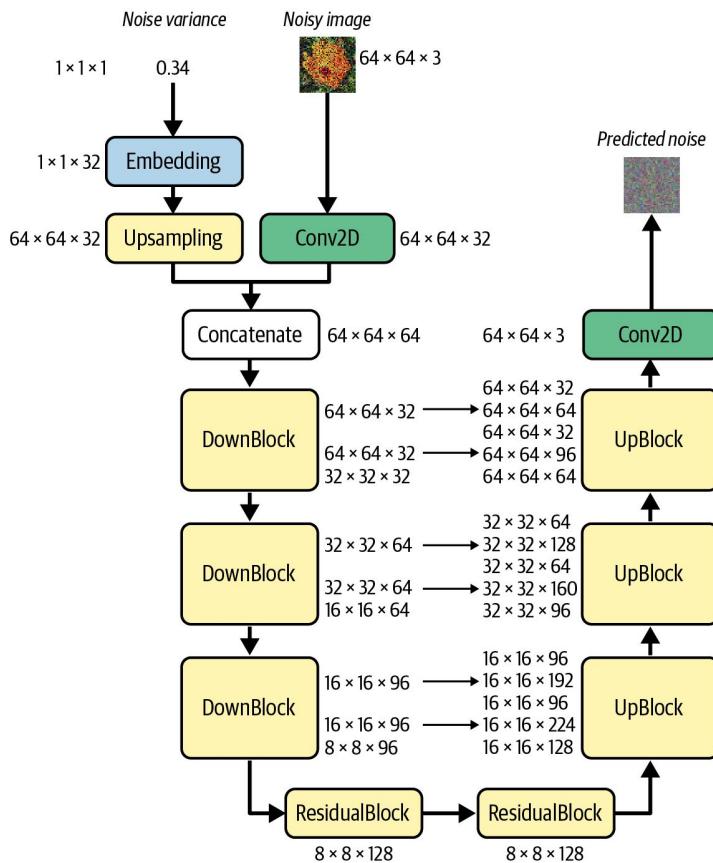


# 4. Modelo: Redes Neurais Profundas

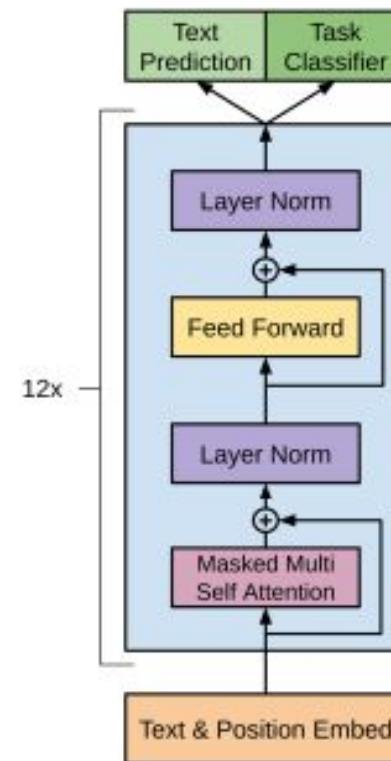
- **Processamento em GPU.**
- **Paralelismo.**
- **Cloud Computing.**
- GPT-4 foi treinado com 25 mil GPUs Nvidia A100.



**Modelos atuais de imagem** como DALL-E, Midjourney e Stable Diffusion são **modelos de difusão baseados na arquitetura U-Net**



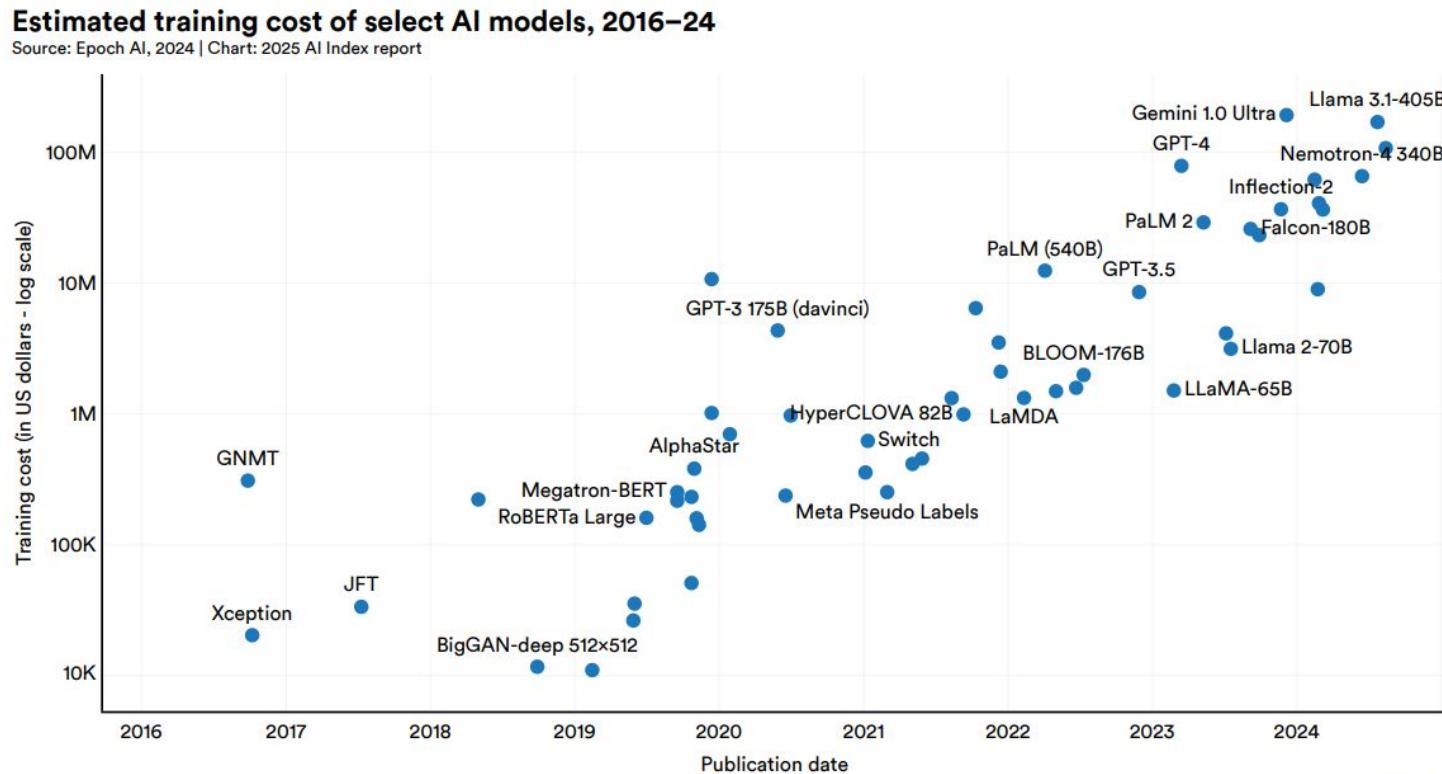
**Modelos atuais de texto** como GPT4, Llama e Deepseek R1 são **modelos autoregressivos baseados na arquitetura Transformer**.



*Abordaremos essas arquiteturas em aulas posteriores!*

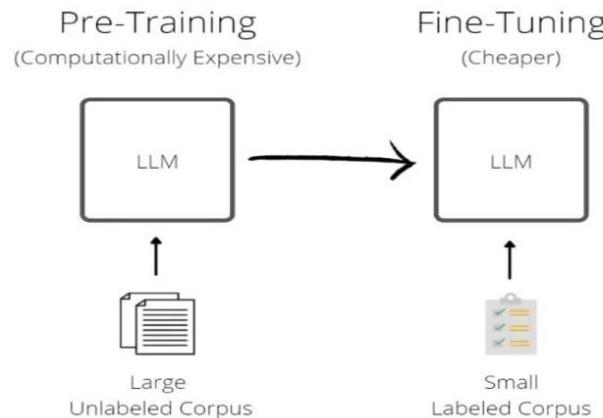
# Modelos geratitivos são caros!

- O treinamento do GPT-3 custou 4.6mi USD, do GPT-4 custou 68mi USD.
- Modelos geratitivos são caros e não podem ser re-treinados a qualquer momento.
- Como desenvolver funcionalidades sem re-treinar o modelo?



# Solução: Divisão em duas etapas

- **Pre-training:** Parte mais intensiva do treinamento, treinamento com um dataset maior não-estruturado de forma **auto-supervisionada**.
  - O output desta etapa é um modelo gerativo genérico, sem funções bem estabelecidas, denominado **modelo de fundação**.
- **Fine-tuning:** Parte menos intensiva, **transfer learning** é aplicado com um dataset menor rotulado por especialistas, organizado para funções específicas como Q&A ou correção de código.
  - Aplicações diferentes podem ter o mesmo modelo fundacional.
  - Ex: ChatGPT e Microsoft Copilot utilizam o GPT-4 como fundacional, mas são tunados para tarefas diferentes.

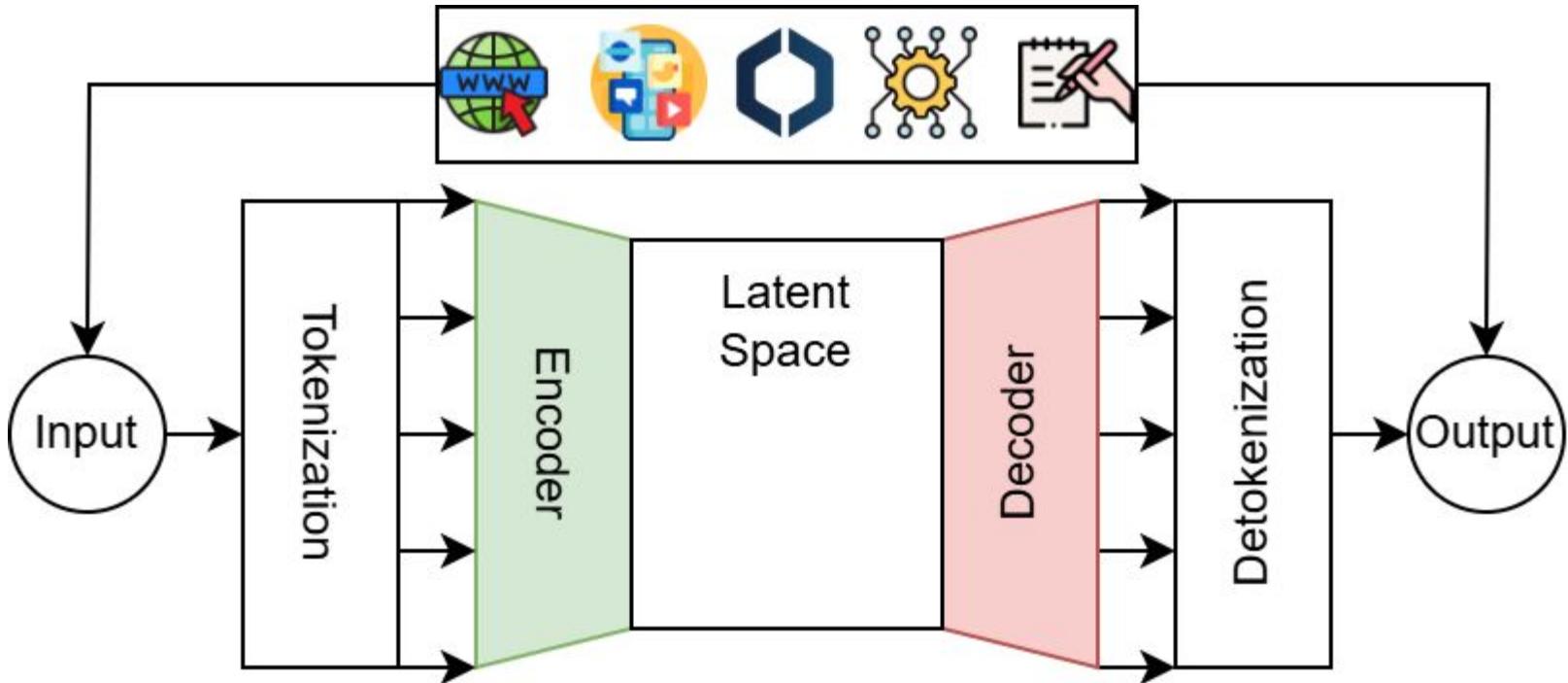


# 1. Exemplo: OpenOrca

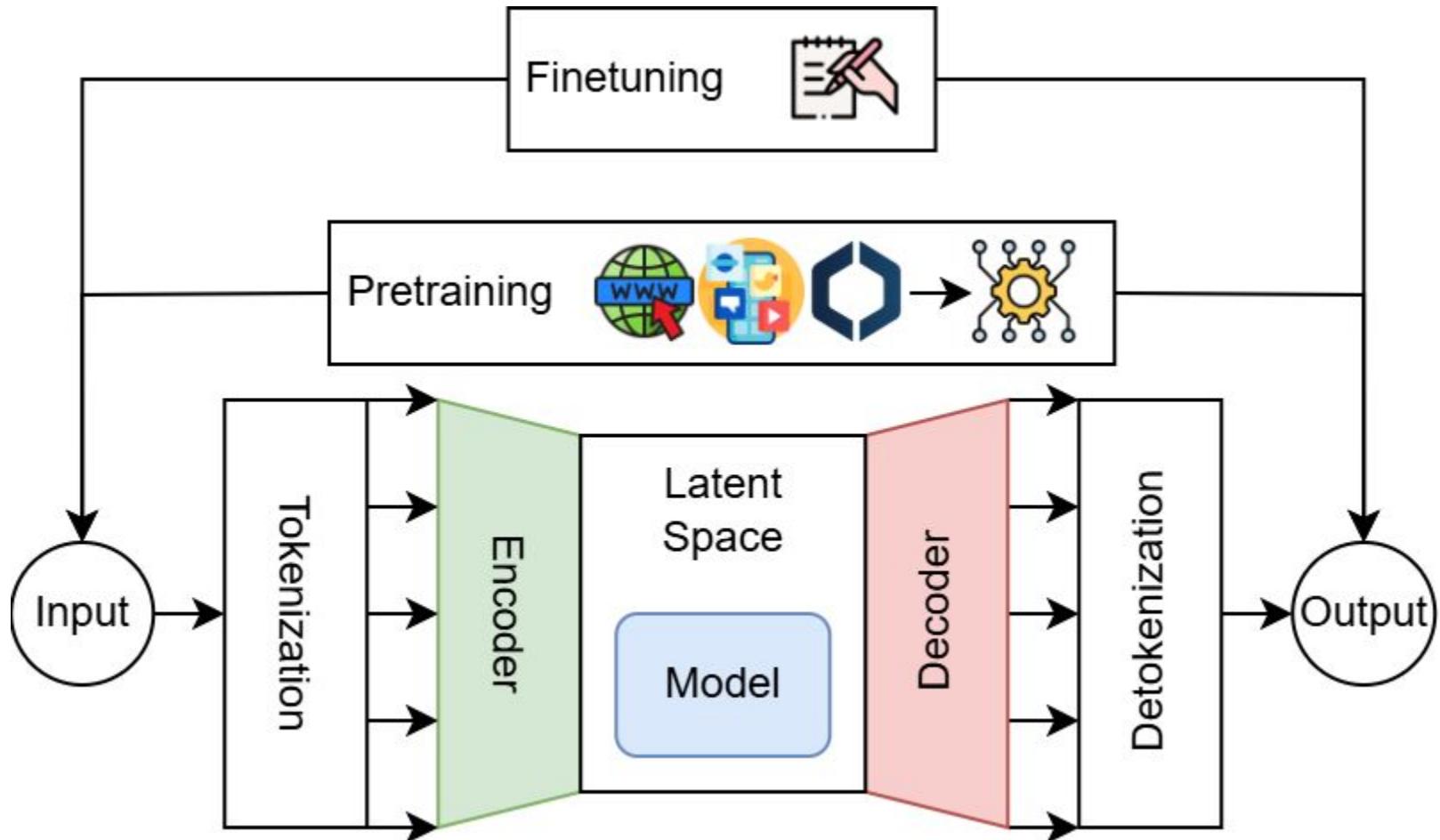
- O [OpenOrca](#) é um dataset Open-Source desenvolvido para tunar um modelo de linguagem.
- Dataset no formato Q&E:
  - Utilizado para desenvolver as capacidades típicas de um chatbot.



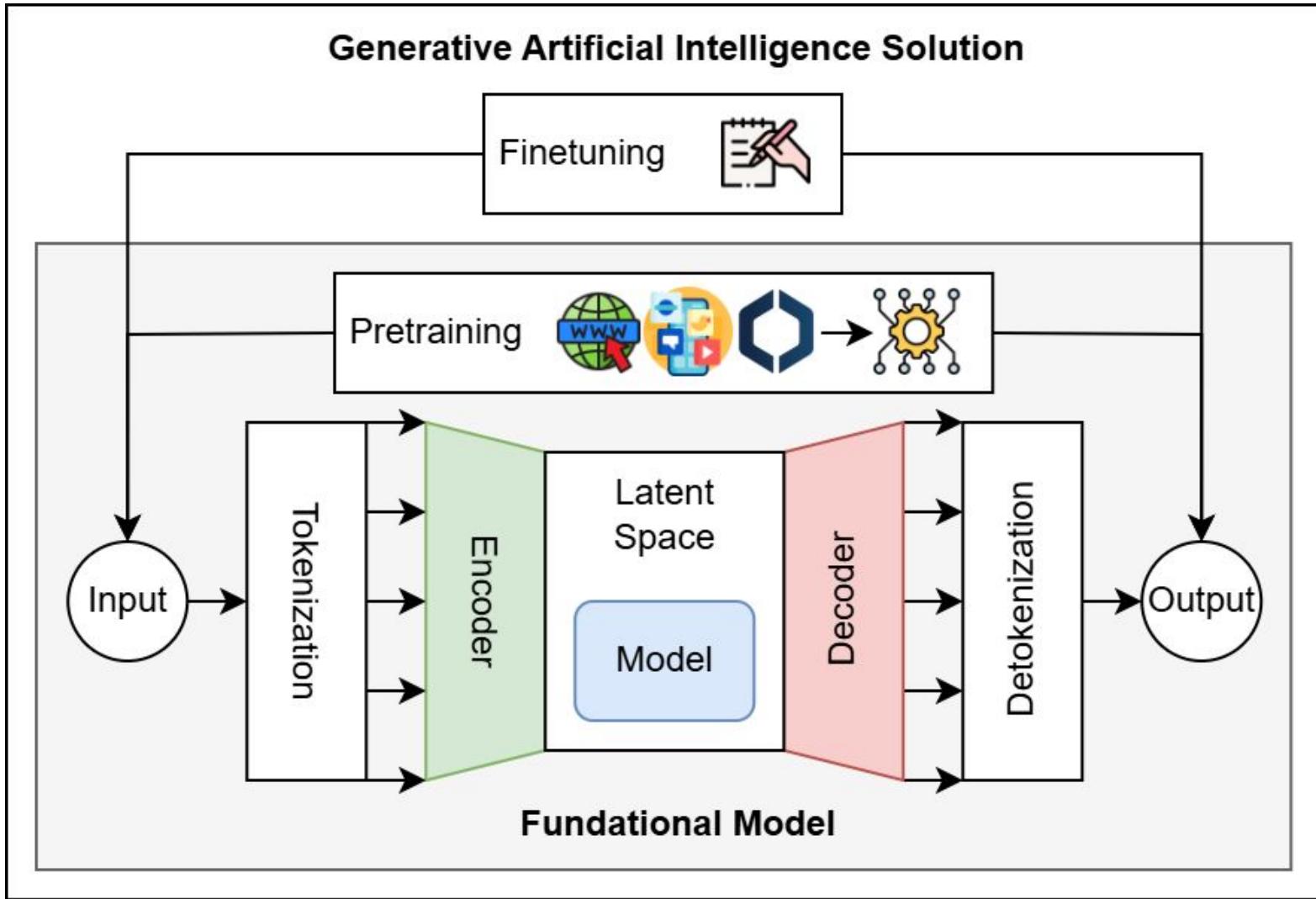
# Modelo: *Pre-training* e *fine-tuning*



# Modelo: *Pre-training* e *fine-tuning*



# Temos uma GenAI.



# Referências

- BRAINS. **Embeddings: medidas de distância e similaridade.** 2024. Disponível em: <https://brains.dev/2024/embeddings-medidas-de-distancia-e-similaridade/>. Acesso em: 26 maio 2025.
- BRAINS. **Token e embedding: conceitos da IA e LLMs – Parte 1.** 2024. Disponível em: <https://brains.dev/2024/token-e-embedding-conceitos-da-ia-e-langs/>. Acesso em: 26 maio 2025.
- BRAINS. **Modelos fundacionais: modelos que revolucionaram a IA.** 2023. Disponível em: <https://brains.dev/2023/foundation-models-modelos-que-revolucionaram-a-ia/>. Acesso em: 26 maio 2025.
- FOSTER, David. **Generative Deep Learning.** 2. ed. O'Reilly Media, 2022. Repositório: [https://github.com/davidADSP/Generative\\_Deep\\_Learning\\_2nd\\_Edition](https://github.com/davidADSP/Generative_Deep_Learning_2nd_Edition). Acesso em: 26 maio 2025.
- HESAMATION. **Primer on LLM Embedding.** Hugging Face. Disponível em: [https://huggingface.co/spaces/hesamation/primer-llm-embedding?section=bert\\_\(bidirectional\\_encoder\\_representations\\_from\\_transformers\)](https://huggingface.co/spaces/hesamation/primer-llm-embedding?section=bert_(bidirectional_encoder_representations_from_transformers)). Acesso em: 26 maio 2025.
- NVIDIA. **Deep Learning.** Disponível em: <https://developer.nvidia.com/deep-learning>. Acesso em: 26 maio 2025.
- OPENAI. **Tokenizador do ChatGPT.** Disponível em: <https://platform.openai.com/tokenizer>. Acesso em: 26 maio 2025.
- RASCHKA, Sebastian. **Build a Large Language Model (From Scratch).** Simon and Schuster, 2024.
- RAVINKUMAR. **Image tokenization. Gen AI Guidebook.** Disponível em: [https://ravinkumar.com/GenAiGuidebook/image/image\\_tokenization.html](https://ravinkumar.com/GenAiGuidebook/image/image_tokenization.html). Acesso em: 26 maio 2025.
- SCRIBBLE DATA. **Foundation Models 101: A Step-by-step Guide for Beginners.** Disponível em: <https://www.scribbledata.io/blog/foundation-models-101-a-step-by-step-guide-for-beginners/>. Acesso em: 26 maio 2025.
- WANG, Yiqiu et al. **Image Token Learner for Vision Language Pretraining.** arXiv preprint arXiv:2406.07550, 2024. Disponível em: <https://arxiv.org/abs/2406.07550>. Acesso em: 26 maio 2025.

# Obrigado

**UDESC – Universidade do Estado de  
Santa Catarina**

Thiago Brandenburg

[thiago.brandenburg@edu.udesc.br](mailto:thiago.brandenburg@edu.udesc.br)