

Processamento de Linguagem Natural

Ronierison Maciel

Julho 2025



Quem sou eu?



Nome: Ronierison Maciel / Roni

Formação: Mestre em Ciência da Computação

Ocupação: Pesquisador, Professor e Escovador de bits

Hobbies: Jogar cartas, ficar com a família

Interesses: Carros, DevSec, DS e ML

Email: ronierison.maciel@pe.senac.br

GitHub: <https://github.com/ronierisonmaciel>

- 1 Introdução ao PLN
- 2 Representação de Texto
- 3 Análise Semântica
- 4 Aprendizado de Máquina em PLN

Curso de Extensão em Processamento de Linguagem Natural (PLN)

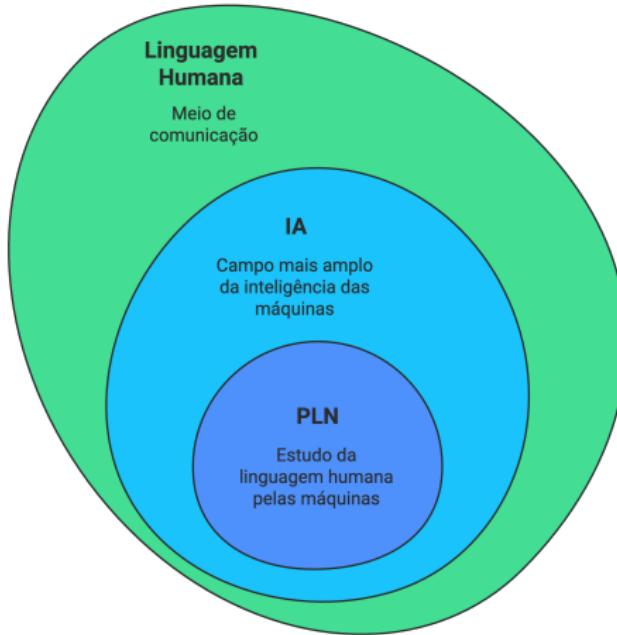
Objetivos:

- História do PLN, desafios, ética e viés
- Tokenização, stopwords, stemming
- Desenvolver os primeiros scripts de PLN

Pergunta: Onde vocês já viram PLN ser aplicado?

Compreender o que é PLN

O que é PLN?



Objetivo: Fazer o computador "entender" linguagem natural.

Conhecer a História e Aplicações



O início do Deep Learning marca um novo capítulo na IA



A revolução estatística dos anos 90 transforma o PLN – 1993



Chatbots antigos são exibidos, mostrando a evolução da IA conversacional – 1980



Teorias como a de Schank e experimentos como SHRDLU são explorados – 1969



Os primeiros sistemas de diálogo como ELIZA e PARRY são desenvolvidos – 1966



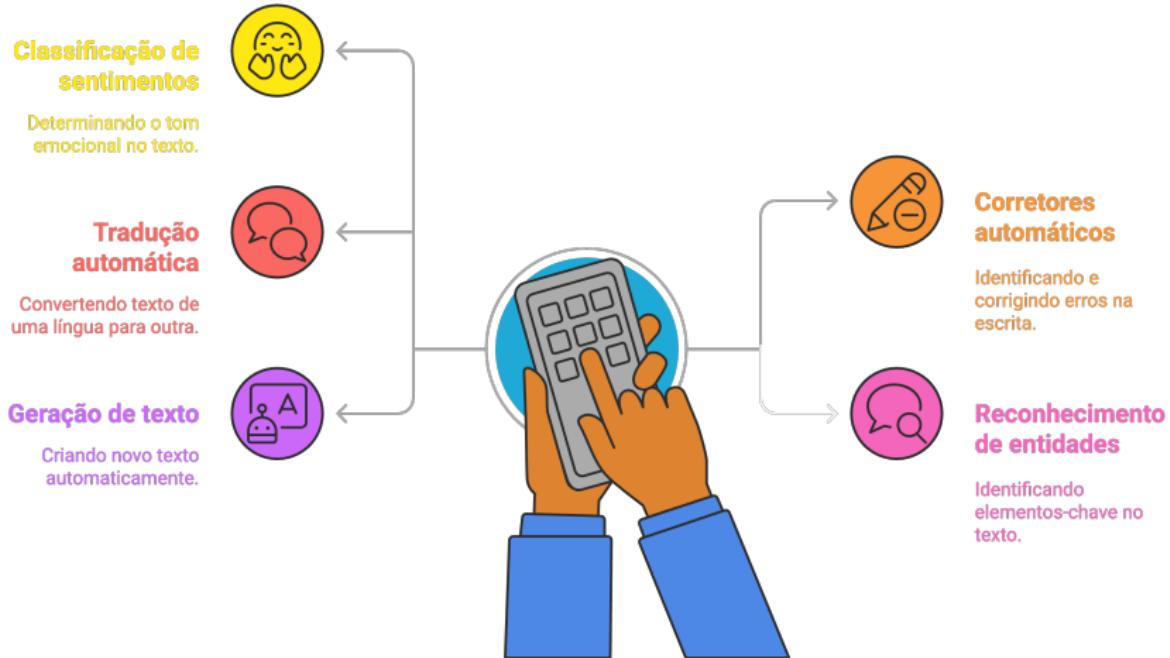
A primeira tradução automática revoluciona a comunicação – 1954



O teste de inteligência de Turing marca o início da IA – 1950



Aplicações de PLN

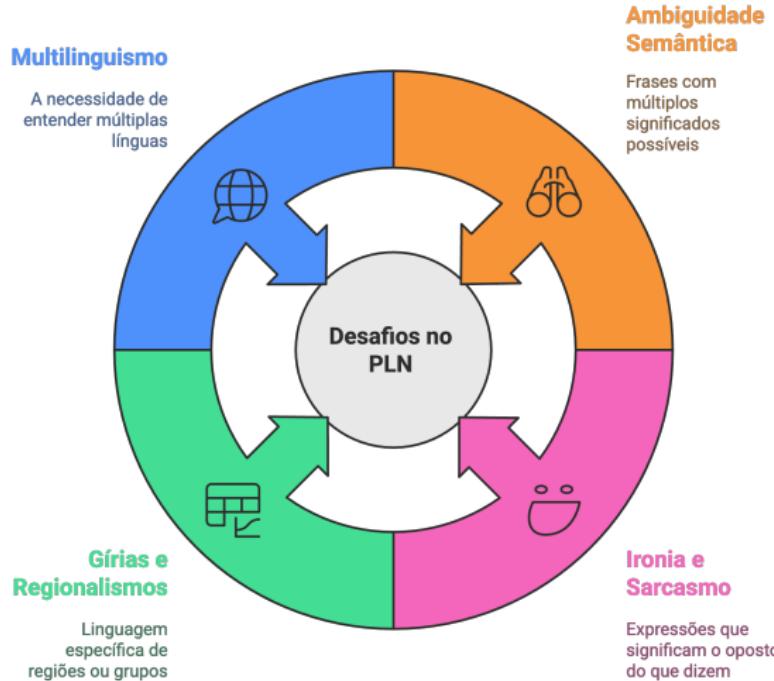


Ciclo do Processamento de Linguagem Natural



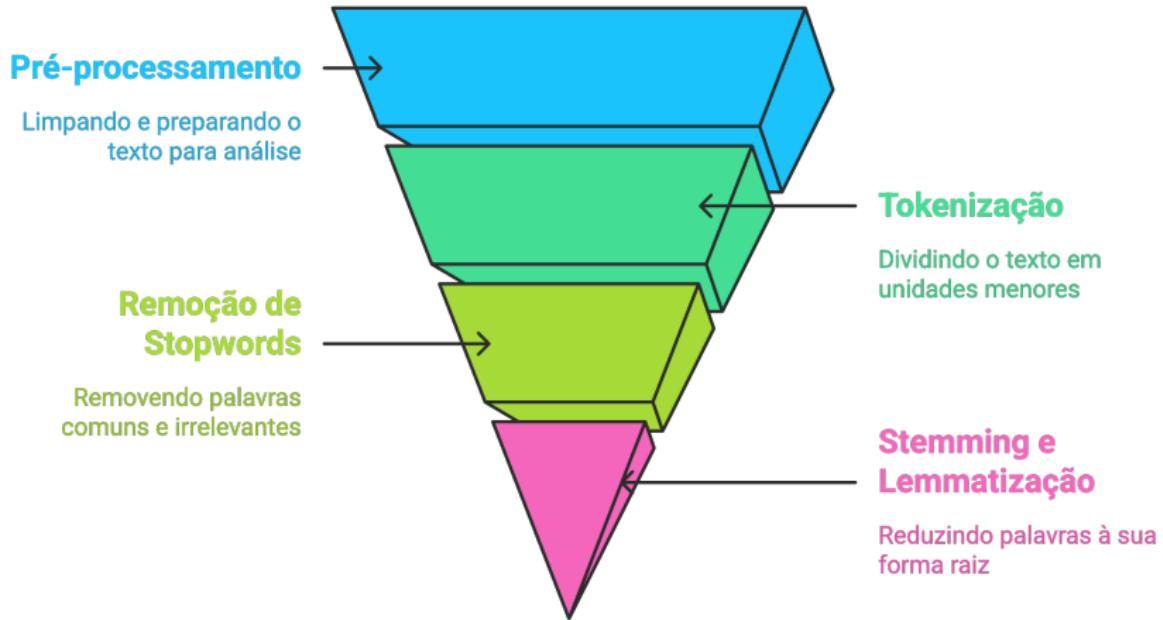
Vocês já usaram algum chatbot que "entendeu errado" sua intenção?

Desafios do PLN



"Esse celular é uma bomba." – Positivo ou negativo?

Processamento de Texto



Mão na massa!

Preparação do Ambiente

Quais ferramentas e recursos precisamos para começar a trabalhar com PLN?

- ① Google Colab ou VS Code
- ② Jupyter
- ③ SpaCy
- ④ NLTK

```
1 # bilbioteca Python  
2 %pip install nltk spacy
```



Importação de bibliotecas

```
1 import nltk
2 import spacy
3
4 # O necessário para esta aula
5 from nltk.tokenize import word_tokenize
6 from nltk.corpus import stopwords
7 nltk.download('punkt_tab')
8 nltk.download('stopwords')
```



Texto exemplo para Análise

```
1 texto = "Mais vale um asno que me carregue que um cavalo que  
    me derrube."  
2 print(texto)
```



Tokenização com NLTK

```
1 tokens_nltk = word_tokenize(texto, language='portuguese')
2 print(tokens_nltk)
```



Remoção de Stopwords

```
1 stopwords_pt = stopwords.words('portuguese')
2 filtered = [w for w in tokens_nltk if w not in stopwords_pt]
3 print(filtered)
```

O que são Stopwords?

Palavras comuns em textos, como "o", "é", "sou", que têm pouco valor semântico. Elas são filtradas no pré-processamento para melhorar a eficiência e a precisão nas tarefas de PLN.

Stemming com NLTK

```
1 from nltk.stem import PorterStemmer  
2 stemmer = PorterStemmer()  
3  
4 stems = [stemmer.stem(word) for word in filtered]  
5 print(stems)
```

O que é Stemming?

Técnica que reduz palavras às suas raízes (radicais), cortando sufixos. Por exemplo, “amando” e “amarei” viram “am”.

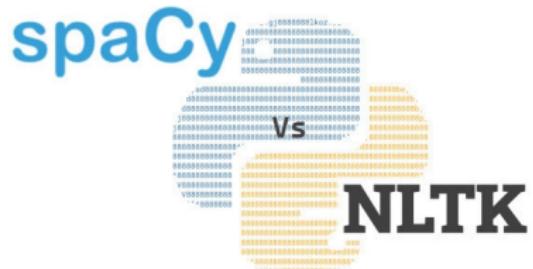
Obs.: o radical pode não ser uma palavra correta e não é a mesma coisa que o lema.

SpaCy: Tokenização e Lematização

```
1 # Instalação da lib pt_core_news_sm
2 !python -m spacy download pt_core_news_sm
3
4 nlp = spacy.load("pt_core_news_sm")
5
6 doc = nlp(texto)
7
8 for token in doc:
9     print(f"{token.text[:10]} | {token.lemma_[:10]} | {token.
pos_[:10]})
```

Qual diferença vocês perceberam entre spaCy e NLTK?

- O spaCy já traz muito embutido: POS, lematização, dependências, etc.
- O NLTK precisa de funções manuais para cada tarefa.



Teste você mesmo!

```
1 texto = "O preço do café avança rapidamente, isso é  
     preocupante."  
2 tokens = word_tokenize(texto.lower())  
3 tokens_filtrados = [w for w in tokens if w not in  
     stopwords_pt]  
4 stems = [stemmer.stem(word) for word in tokens_filtrados]  
5 print("Original:", texto, "\n\nTokens:", tokens, "\n\nSem  
     stopwords:", tokens_filtrados, "\n\nStems:", stems)
```

Teste seus conhecimentos!



Conceitos vistos hoje:

- Introdução ao PLN e aplicações reais
- Etapas iniciais do processamento de texto
- Tokenização, remoção de stopwords e stemming
- Prática

Tarefa: Traga um exemplo de uso de PLN que você viu ou utiliza no dia a dia.

Objetivo:

- Representação de texto + modelagem de tópicos + embeddings
 - Representações vetoriais (BoW, TF-IDF, Word2Vec)
 - Gensim (LDA) + Scikit-learn para vetores e embeddings
 - Gensim, Scikit-learn, spaCy
 - Classificador de sentimento com base em vetor

Objetivo:

- Transformar texto em números para uso em modelos.
- Principais modelos:
 - Bag of Words (BoW)
 - TF-IDF
 - Word2Vec

Exemplo prático: "Eu gosto de Café" → [1, 1, 1, 1, 0, 0, ...]

Qual modelo de representação vetorial usar para análise de texto?

Bag of Words

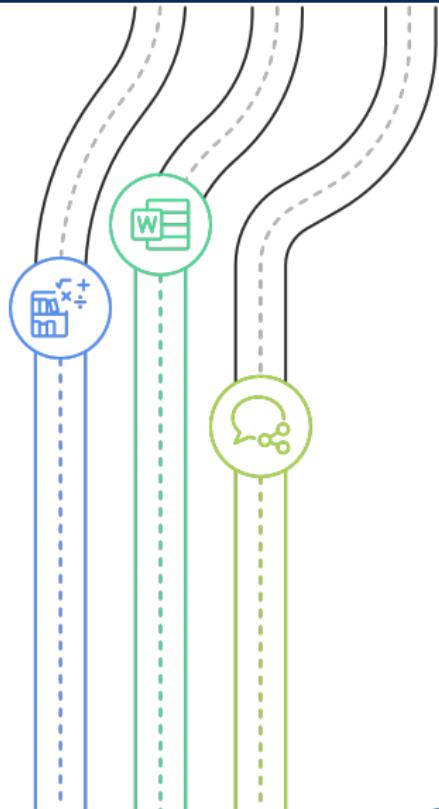
Conta a frequência das palavras, simples mas pode perder contexto.

TF-IDF

Pesa palavras com base na singularidade, útil para identificar palavras-chave.

Word2Vec

Captura o significado das palavras com base no contexto, mais complexo mas preciso.



Importação de bibliotecas

Quais ferramentas e recursos precisamos?

- ① Scikit-learn
- ② Gensim

```
1 # Instalar as bibliotecas necessárias
2 %pip install -U scikit-learn spacy gensim
3
4 # Carregar o modelo pt_core_news_sm
5 !python -m spacy download pt_core_news_sm
```

Pré-processamento com spaCy

```
1 import spacy  
2  
3 nlp = spacy.load("pt_core_news_sm")  
4  
5 texto = "O café nos permite desenvolver script e obter  
       conhecimentos computacionais."  
6 doc = nlp(texto)  
7  
8 for token in doc:  
9     print(f"{token.text:<12} | {token.lemma_:<12} | {token.  
pos_:<10}")
```

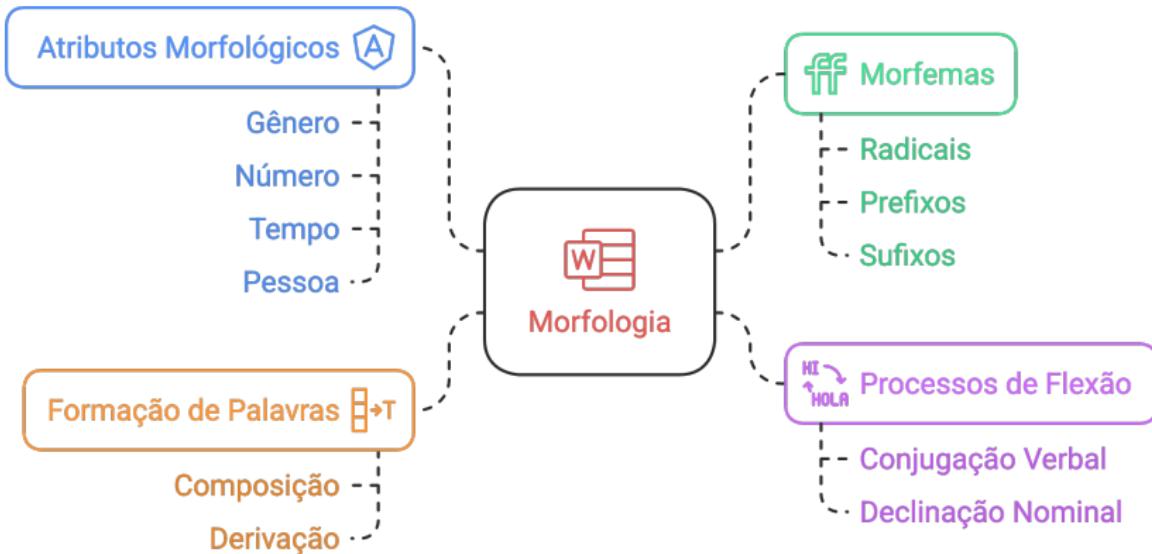


Análise Morfossintática

Token	Lema	Classe (POS)
o	o	DET
café	café	NOUN
nos	nós	PRON
permite	permitir	VERB
desenvolver	desenvolver	VERB
script	script	ADJ
e	e	CCONJ
obter	obter	VERB
conhecimentos	conhecimento	NOUN
computacionais	computacional	ADJ
.	.	PUNCT

Table: Tokens, lemas e classes gramaticais

Morfologia linguística



Bag of Words com Scikit-learn

```
1 from sklearn.feature_extraction.text import CountVectorizer  
2  
3 corpus = ["gosto de Café", "Café é legal"]  
4 vectorizer = CountVectorizer()  
5 X = vectorizer.fit_transform(corpus)  
6  
7 print("Vocabulário:", vectorizer.get_feature_names_out())  
8 print("Matriz BoW:", X.toarray())
```

Matriz Bag of Words

O modelo BoW transforma um texto em uma matriz de ocorrência de palavras. Vamos usar **CountVectorizer** para aplicar BoW em um corpus simples.

Palavra	Linha 1 ("gosto de café")	Linha 2 ("café é legal")
café	1	1
de	1	0
gosto	1	0
legal	0	1

Table: Representação Bag of Words

TF-IDF com Scikit-learn

O TF-IDF calcula a importância de cada palavra no documento levando em conta sua frequência inversa no corpus.

```
1 from sklearn.feature_extraction.text import TfidfVectorizer  
2  
3 vectorizer = TfidfVectorizer()  
4 X = vectorizer.fit_transform(corpus)  
5  
6 print("Vocabulário:", vectorizer.get_feature_names_out())  
7 print("Matriz TF-IDF:", X.toarray())
```

Matriz TF-IDF

- **café** aparece em ambas as frases, por isso seu peso TF-IDF é moderado nas duas linhas. É uma palavra frequente no corpus, então seu IDF não é tão alto.
- **de e gosto** só aparecem na linha 1, o que gera um peso maior para elas nessa sentença. Como não aparecem em outros documentos, seu IDF é alto.
- **legal** só aparece na linha 2, e também tem peso alto por esse motivo.

Palavra	Linha 1 ("gosto de café")	Linha 2 ("café é legal")
café	0.45	0.58
de	0.63	0.00
gosto	0.63	0.00
legal	0.00	0.81

Table: TF-IDF: Importância relativa das palavras por sentença



Discussão rápida

Pergunta:

- O que muda entre BoW e TF-IDF?
- Qual seria melhor para classificar sentimentos?

Atividade: Altere as frases no corpus e observe os vetores gerados.

Word2Vec com Gensim

Word2Vec aprende a representação semântica das palavras com base em seus contextos. Aqui usamos um corpus simples para ilustrar.

```
1 from gensim.models import Word2Vec  
2  
3 frases = [["Café", "é", "legal"], ["texto", "vetorial", "modelo"]]  
4 model = Word2Vec(sentences=frases, vector_size=10, min_count=1)  
5  
6 print("Vetor de 'Café':", model.wv["Café"])  
7 print("Palavras similares a 'Café':", model.wv.most_similar("Café"))
```

Vetor da palavra café

- Valores positivos (em verde): indicam dimensões com maior relevância semântica para a palavra.
- Valores negativos (em vermelho): mostram menor influência ou sentido oposto.

Dimensão	Valor
1	-0.0816
2	0.0450
3	-0.0414
4	0.0082
5	0.0850
6	-0.0446
7	0.0452
8	-0.0679
9	-0.0355
10	0.0940

Palavras mais similares a café

A tabela mostra as palavras próximas de "café" no espaço vetorial, de acordo com a similaridade de cossenos.

- Os valores positivos (em verde) indicam que as palavras aparecem em contextos semelhantes (ex: "é", "legal").
- Os valores negativos (em vermelho) sugerem que essas palavras ocorrem em contextos diferentes ou opostos (ex: "modelo", "vetorial").

Palavra	Similaridade
é	0.2495
legal	0.0927
texto	-0.1517
modelo	-0.2726
vetorial	-0.3821

Reflexão sobre embeddings

Perguntas:

- Quais palavras ficaram "próximas"?
- A ordem das palavras influencia nos vetores?

Desafio: Monte seu próprio conjunto de frases e explore vetores.

Modelagem de Tópicos com LDA

Objetivo: Identificar automaticamente os assuntos principais em um conjunto de textos.

Exemplo:

- Tópico 1: "tecnologia", "IA", "dados"
- Tópico 2: "vacina", "saúde", "covid"

Exemplo de LDA com Gensim

LDA (Latent Dirichlet Allocation) é um método de modelagem de tópicos que identifica temas ocultos nos documentos.

```
1 from gensim import corpora, models  
2  
3 docs = [["inteligência", "computacional"], ["Café", "texto",  
        "dados"]]  
4 dicionario = corpora.Dictionary(docs)  
5 corpus = [dicionario.doc2bow(doc) for doc in docs]  
6  
7 lda = models.LdaModel(corpus, num_topics=2, id2word=  
        dicionario, passes=10, iterations=50)  
8 for idx, topic in lda.print_topics():  
     print(f"Tópico {idx}: {topic}")
```

Tópicos extraídos pelo modelo LDA

- O **tópico 0** agrupa principalmente as palavras "computacional" e "inteligência", indicando um tema relacionado a inteligência computacional.
- O **tópico 1** destaca "texto", "dados" e "Café", sugerindo um tema mais relacionado a manipulação de dados e texto, incluindo a palavra "Café".

Tópico	Palavras e pesos
0	0.330 " <i>computacional</i> " + 0.330 " <i>inteligência</i> " + 0.113 " <i>Café</i> " + 0.113 " <i>dados</i> " + 0.113 " <i>texto</i> "
1	0.271 " <i>texto</i> " + 0.271 " <i>dados</i> " + 0.271 " <i>Café</i> " + 0.093 " <i>inteligência</i> " + 0.093 " <i>computacional</i> "

Comparação: Processamento sem e com spaCy

Técnica	Sem spaCy	Com spaCy
BoW / TF-IDF	Tem ruído, palavras inúteis, redundância	Foco no essencial (menos colunas, mais sinal)
Word2Vec	Treina em dados redundantes e confusos	Vetores limpos, sem stopwords
LDA	Tópicos com palavras soltas e sem valor	Tópicos coerentes e sem ruído gramatical

Table: Efeito do pré-processamento com spaCy nas representações textuais

Atividade prática – LDA

Desafio:

- Colete 5 a 10 notícias pequenas
- Faça pré-processamento com spaCy (tokens + stopwords)
- Execute LDA e interprete os tópicos

Objetivo: Construir um modelo simples que detecta sentimento positivo ou negativo com BoW ou TF-IDF.

Etapas:

- Criar dataset de frases e sentimentos
- Vetorizar os textos com Scikit-learn
- Treinar com Naive Bayes
- Avaliar o resultado com métricas simples

Exemplo inicial do mini-projeto

```
1 from sklearn.naive_bayes import MultinomialNB  
2  
3 frases = ["gosto de PLN", "detesto filas"]  
4 y = ["positivo", "negativo"]  
5  
6 vectorizer = CountVectorizer()  
7 X = vectorizer.fit_transform(frases)  
8  
9 modelo = MultinomialNB().fit(X, y)  
10 print(modelo.predict(vectorizer.transform(["PLN é ótimo!"])))  
    )
```



Resumo

Resumo:

- Representações vetoriais de texto (BoW, TF-IDF, Word2Vec)
- Modelagem de tópicos com LDA
- Uso das bibliotecas Gensim, Scikit-learn e spaCy
- Mini-projeto de classificação de sentimento

Para casa: Aprimore seu modelo com mais frases e teste em comentários reais da internet.

Objetivos de hoje

- BERT, GPT e arquitetura transformer
- Zero-shot classification, NER, QA com Hugging Face
- Ferramenta ativa: transformers, datasets, torch
- Geração de texto com modelo pré-treinado

Evolução da Arquitetura Transformer



O Transformer se torna a base para arquiteturas modernas de linguagem natural.



Modelos como BERT e GPT são construídos sobre a arquitetura Transformer.



A arquitetura é composta por blocos encoder e decoder para processamento e geração.



O modelo captura dependências de longo alcance entre palavras, melhorando a compreensão.



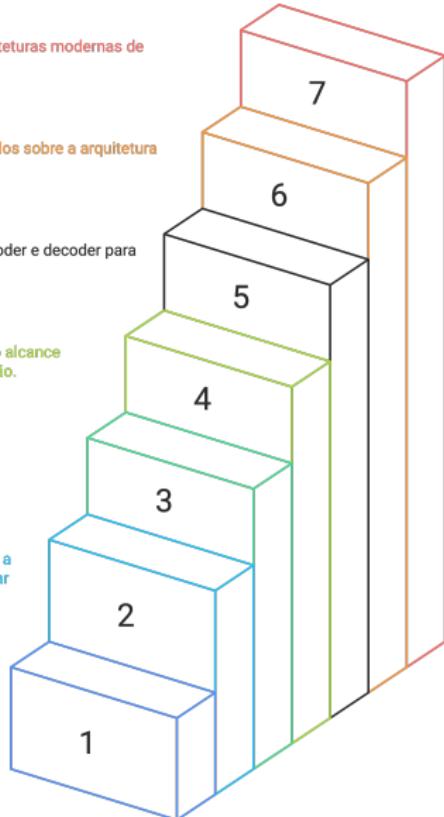
A arquitetura permite paralelização no treinamento, melhorando a eficiência.



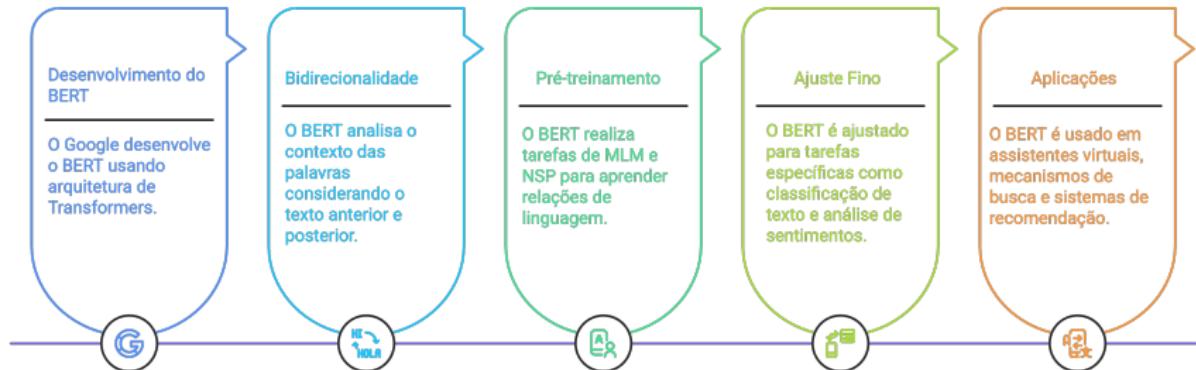
O mecanismo de atenção, especialmente a atenção multi-cabeça, é usado para avaliar relações entre palavras.



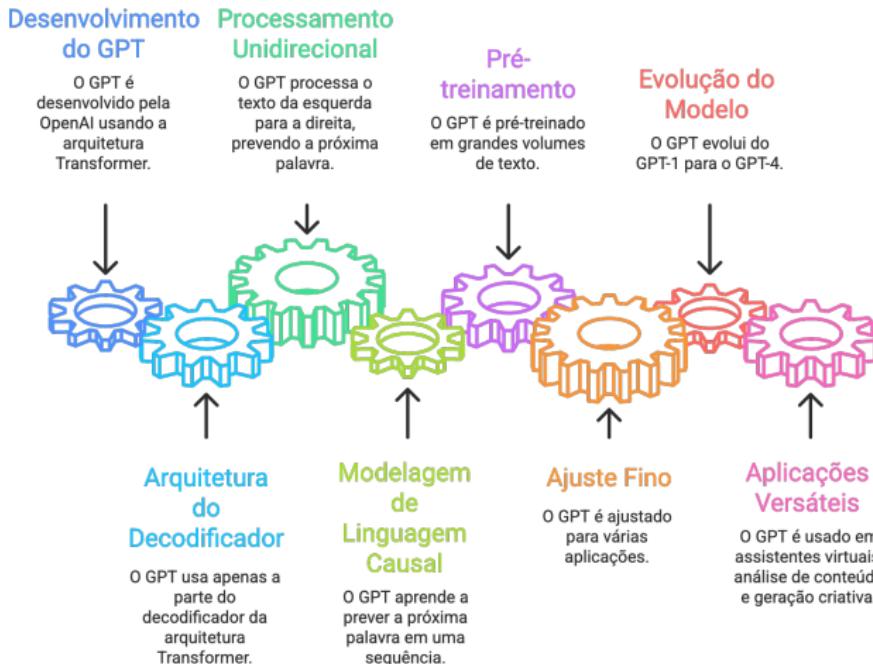
A arquitetura Transformer é introduzida por Vaswani et al. em 2017.



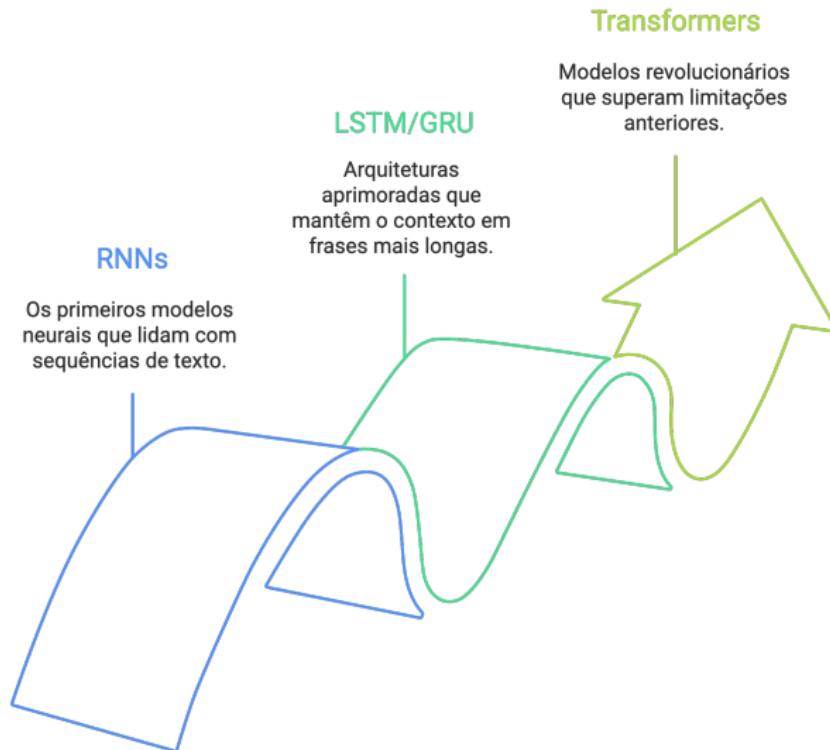
Processo de Desenvolvimento e Aplicações do BERT



Processo do Modelo GPT



Evolução dos modelos neurais em PLN

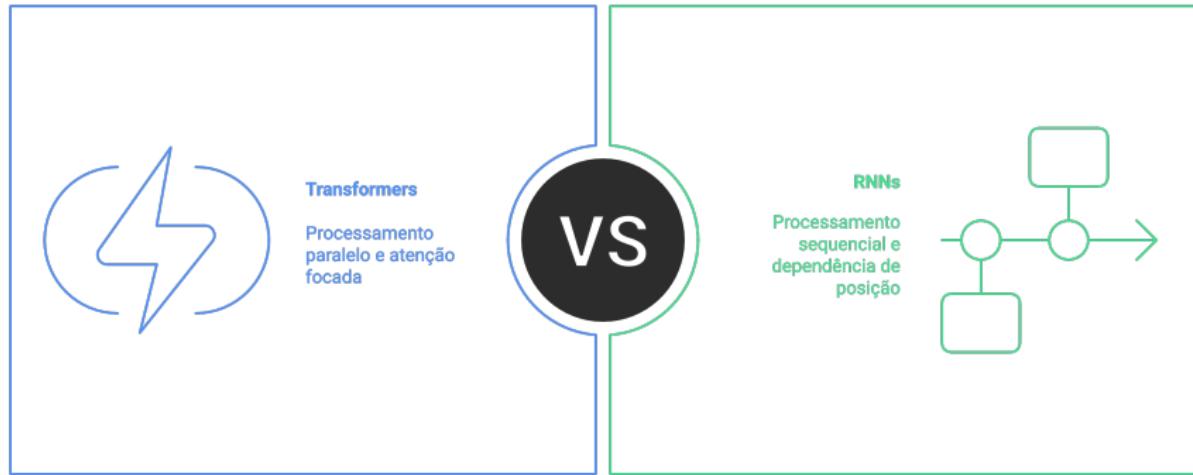


Limitações das Redes Neurais Recorrentes



Transformers: a Revolução

Qual abordagem de modelo neural é mais eficiente para o processamento de texto?



Inovação dos Transformers

Quais são as duas inovações cruciais dos Transformers?

Mecanismo de Atenção e Leitura Paralela.

O que é o Mecanismo de Atenção?

Permite ao modelo focar nas palavras mais relevantes de uma frase, independentemente da posição em que estão.

O que é Leitura Paralela?

Ao contrário das RNNs, os Transformers processam todo o texto de uma vez, tornando o treinamento muito mais rápido e eficiente.



Modelos baseados em Transformers

Comparação entre BERT, GPT e T5

Característica	BERT	GPT	T5
 Compreensão do contexto	Contexto bidirecional	Contexto unidirecional	Conversão de texto para texto
 Geração de texto	Geração limitada	Autônoma e fluente	Conversão de texto para texto
 Abordagem da tarefa	Compreensão contextual	Generativa	Conversão de texto

Arquiteruras de PLN



BERT

Leitura bidirecional,
ótimo para tarefas de
classificação.

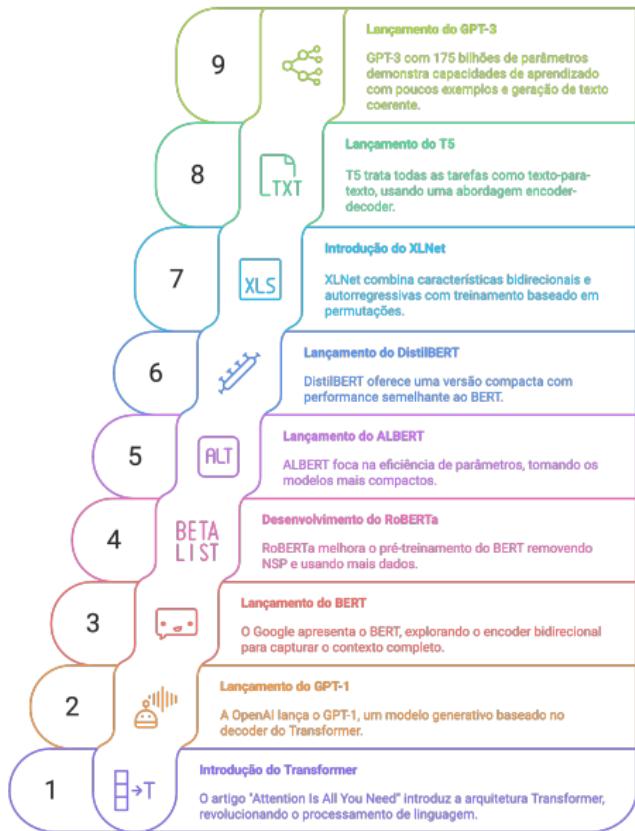
Geração de texto
autoregressiva,
excelente para criação
de conteúdo.



Outros

Inclui os modelos T5,
RoBERTa, DistilBERT e
BART.

Evolução dos Modelos Transformer



Simulações de Modelos Transformers

A seguir, faremos algumas simulações de Modelos Transformers, para uma compreensão mais sólida:

- **BERT**,
- **GPT** e
- **T5**

BERT: predição de uma palavra no meio da frase

O notebook completo está no repositório PLN MBA

```
1 frase_mascarada = ["0", "aluno", "foi", "a", "[MASK]", "para"
2   , "estudar"]
3
4 candidatos = {
5     "biblioteca": 0.85,
6     [...]
7 }
8
9 escolhida = max(candidatos, key=candidatos.get)
10 [...]
11 print("Frase final:", " ".join(frase_final))
12 [...]
```



Alinhado com a realidade.

- O BERT vê todo o contexto (antes e depois da palavra).
- Substitui [MASK] com a palavra mais provável.
- Serve para pré-treinamento de modelos.

GPT – Geração de texto autoregressiva (next word prediction)

O notebook completo está no repositório PLN MBA

```
1 gpt_simulado = {  
2     "Era": {"uma": 0.8, "vez": 0.2},  
3     [...]  
4 }  
5  
6 def escolher(pesos):  
7     pass  
8  
9 prompt = ["Era"]  
10 while len(prompt) < 20:  
11     [...]  
12  
13 print("Texto gerado:", " ".join(prompt))
```

Alinhado com a realidade.

- GPT só olha para o passado (não sabe o futuro).
- Gera palavra por palavra, sempre com base no histórico.
- Ideal para continuação de textos.

T5 – Text-to-text: tarefa definida por instrução textual

O notebook completo está no repositório PLN MBA

```
1 entrada = "resuma: A computação em nuvem permite o acesso  
2     remoto a servidores, armazenamento e bancos de dados  
3     pela internet."  
4  
5 def resumir(texto):  
6     pass  
7  
8 [...]
```

Alinhado com a realidade.

- T5 trata tudo como texto para texto.
- Recebe uma instrução textual do que fazer.
- Pode realizar tradução, resumo, QA, classificação, etc.

Mão na massa!

Agora que já sabemos como os modelos

- BERT,
- GPT e
- T5 se comportam. Vamos em frente...

Pré-requisitos para as práticas: Para que nossos scripts funcionem corretamente, precisamos das seguintes bibliotecas.

- transformers,
- datasets e
- torch

```
1 pip install transformers datasets torch
```

Se estiver em notebook ou Google Colab, funciona direto.

BERT – Preenchendo Máscaras (Masked Language Modeling)

O notebook completo está no repositório PLN MBA

```
1 from transformers import AutoTokenizer, AutoModelForMaskedLM
2 import torch
3
4 tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
5 model = AutoModelForMaskedLM.from_pretrained("bert-base-
6 uncased")
7 [...]
8 print("Frase original:", frase)
9 print("Palavra predita:", predicted_word)
```

GPT – Geração de Texto (Causal Language Model)

O notebook completo está no repositório PLN MBA

```
1 from transformers import AutoModelForCausalLM, AutoTokenizer  
2  
3 tokenizer = AutoTokenizer.from_pretrained("gpt2")  
4 model = AutoModelForCausalLM.from_pretrained("gpt2")  
5  
6 entrada = tokenizer("My car is", return_tensors="pt")  
7 saída = model.generate(**entrada, max_length=20)  
8  
9 print(tokenizer.decode(saída[0], skip_special_tokens=True))
```

Zero-shot Classification com Transformers

```
1 from transformers import pipeline  
2  
3 classifier = pipeline("zero-shot-classification")  
4 resultado = classifier(  
5     "O atendimento foi ótimo e muito rápido!",  
6     candidate_labels=["elogio", "reclamação", "informação"]  
7 )  
8 print(resultado)
```



Teste seus conhecimentos!



Discussão sobre Zero-shot

Reflexão:

- O modelo acertou a intenção da frase?
- Como isso pode ser útil em SAC ou e-mails?

Desafio: Troque a frase por uma mais ambígua. O modelo se confunde?

Atividade orientada:

- Acesse: <https://huggingface.co/models>
- Explore modelos como:
 - bert-base-multilingual-cased,
 - distilbert-base-uncased
- Leia as tarefas suportadas e formatos de entrada

Geração de texto com GPT-2

```
1 from transformers import pipeline  
2  
3 gerador = pipeline("text-generation", model="gpt2")  
4 saída = gerador("A inteligência artificial no futuro",  
    max_length=30)  
5 print(saida[0]["generated_text"])
```

Discussão sobre geração

Pergunta:

- O texto gerado faz sentido?
- Você confiaria em um artigo 100% gerado por IA?

Tópicos abordados:

- ① Fundamentos da arquitetura Transformer
- ② Diferença entre BERT, GPT e afins
- ③ Prática com Zero-shot e geração de texto
- ④ Ferramentas: Hugging Face, Transformers, Torch

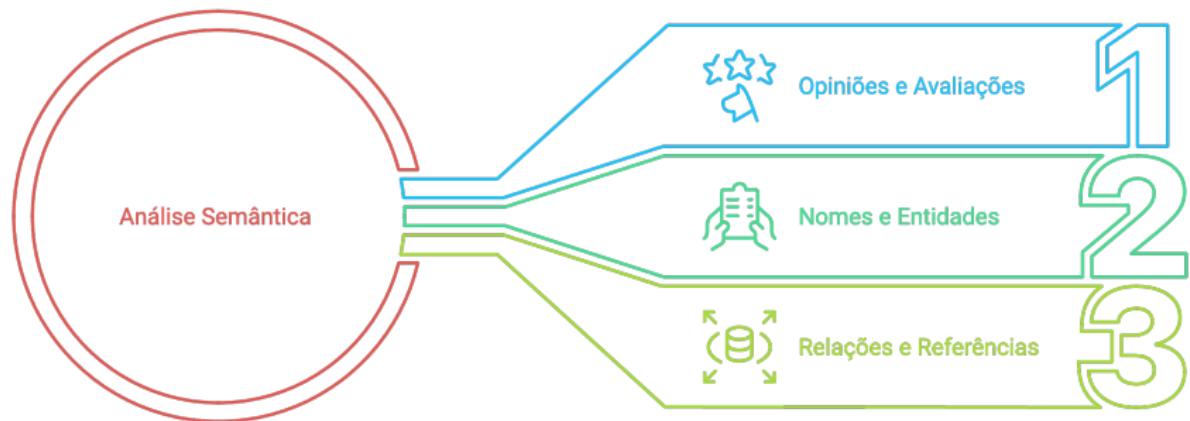
Tarefa: Explore o Hugging Face Spaces e crie um classificador para detectar elogios e reclamações com seus próprios textos.

Objetivos:

- Entender como atribuir significado ao texto
- Explorar análise de sentimento e NER
- Conhecer modelos de linguagem e suas aplicações

Pergunta inicial: Como o computador entende “gostei muito” vs “detestei isso”?

Explorando as Dimensões da Análise Semântica



O que é Análise Semântica?

Ideia central: Entender o significado do texto, não só as palavras.

Aplicações:

- Opiniões e avaliações (review de produtos)
- Nomes e entidades em contratos ou notícias
- Identificar relações e referências no texto

Objetivo: Identificar a polaridade emocional de um texto (**positivo, negativo, neutro**).

Exemplos:

- Avaliações no Google/TripAdvisor
- Comentários em redes sociais

Pergunta: O que torna uma frase "positiva" ou "negativa"?

Exemplo – Análise de Sentimento com TextBlob

```
1 from textblob import TextBlob  
2  
3 frase = TextBlob("I loved the service!")  
4 print(frase.sentiment)
```

Resultado: Polarity varia de -1 (negativo) a +1 (positivo)

Atenção: TextBlob funciona melhor com inglês

Atividade prática

Teste com seus próprios exemplos:

- Pegue 3 frases diferentes
- Aplique o TextBlob e verifique a polaridade
- Tente enganar o sistema com ironia

Reflexão: O modelo conseguiu captar o seu tom?

Reconhecimento de Entidades Nomeadas – NER

Objetivo: Detectar nomes de pessoas, organizações, locais, datas, etc.

Aplicações:

- Extração de dados em contratos
- Identificação de alvos em notícias
- Bases de dados jurídicas ou jornalísticas

Exemplo com spaCy – NER

```
1 import spacy
2 nlp = spacy.load("pt_core_news_sm")
3
4 doc = nlp("Apple compra startup brasileira.")
5 for ent in doc.ents:
6     print(ent.text, ent.label_)
```

Atividade: Crie uma frase com nomes, locais e organizações.

Teste no spaCy:

- A NER detectou corretamente?
- O que ela confundiu ou deixou passar?

Extração de Relações

Ideia: Ir além do reconhecimento. Descobrir ligações entre entidades.

Exemplo: "João trabalha na Google." → (João, trabalha, Google)

Discussão: Como um computador poderia representar essas conexões?

Resolução de Correferências

Problema comum: "Maria comprou um carro. Ela adorou o veículo."

Objetivo: Descobrir a que ou a quem os pronomes se referem.

Desafios:

- Ambiguidade
- Contexto depende de frases anteriores

Definição: Capacidade de prever a próxima palavra ou gerar texto com coerência.

Três tipos principais:

- N-gramas
- Modelos de Markov
- Modelos neurais (RNN, Transformer)

Modelo de N-grama Simples

```
1 from nltk import bigrams
2 from collections import Counter
3
4 texto = "o gato pulou o muro"
5 tokens = texto.split()
6 bigrama = list(bigrams(tokens))
7 contagem = Counter(bigramma)
8 print(contagem)
```

Frequência de Bigramas

Bigramas	Ocorrências
o gato	1
gato pulou	1
pulou o	1
o muro	1

Atividade com N-gramas

Desafio:

- Teste com diferentes frases
- Monte os bigramas e trigramas
- Veja padrões que surgem

Discussão: Como um modelo de n-gramas ajudaria a prever palavras?

Conceito: Probabilidade de uma palavra depende do estado anterior.

Exemplo: "Estou com" → "fome" ou "sono"?

Observação: Base para modelos probabilísticos de linguagem simples.

RNN:

- Aprende sequência com memória recorrente
- Problemas com longas dependências

Transformers:

- Paralelismo e atenção
- Base dos modelos modernos: BERT, GPT

Exemplo prático com HuggingFace (teórico)

Demonstração (via Colab):

- Geração de texto com 'transformers' da HuggingFace
- Entrada: "A inteligência artificial vai..."
- Resultado: texto gerado com coerência

Link sugerido: <https://huggingface.co/spaces>

Resumo do Encontro

Conceitos abordados:

- Análise de sentimento
- Reconhecimento de entidades (NER)
- Extração de relações e correferência
- Modelagem de linguagem: n-gramas, Markov, RNN e Transformers

Tarefa sugerida: Use o spaCy para analisar uma notícia completa. Identifique entidades e tente aplicar extração de relações.

Objetivos:

- Aplicar aprendizado de máquina ao PLN
- Conhecer aplicações modernas como tradução e geração de texto
- Refletir sobre ética, viés e privacidade

Pergunta inicial: Você confiaria em uma IA para escrever um parecer jurídico?

Definição: Atribuir uma categoria a um texto (ex.: positivo ou negativo, spam ou não-spam).

Aplicações:

- Análise de sentimentos
- Detecção de fake news
- Organização de e-mails

Classificação com Naive Bayes

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 from sklearn.naive_bayes import MultinomialNB
3
4 textos = ['Gosto de PLN', 'Não gosto de filas']
5 rotulos = ['positivo', 'negativo']
6 vetor = CountVectorizer()
7 X = vetor.fit_transform(textos)
8
9 clf = MultinomialNB().fit(X, rotulos)
10 print(clf.predict(vetor.transform(['PLN é ótimo!'])))
```

Atividade prática – Classificação

Desafio:

- Crie um conjunto de frases (pelo menos 6)
- Classifique como positivo ou negativo
- Treine e teste seu próprio classificador

Discussão: Seu modelo acertou bem? E se a frase for ambígua?

Clusterização

Objetivo: Agrupar textos semelhantes sem rótulo prévio.

Aplicações:

- Agrupamento de notícias
- Segmentação de clientes
- Descoberta de tópicos

Exemplo – KMeans para Textos

```
1 from sklearn.cluster import KMeans  
2  
3 textos = ['gato dorme', 'cachorro late', 'carro acelera']  
4 X = vetor.fit_transform(textos)  
5  
6 modelo = KMeans(n_clusters=2).fit(X)  
7 print(modelo.labels_)
```

Discussão sobre Clusterização

Reflexão:

- Os textos semelhantes foram agrupados?
- Se mudar uma palavra, muda o grupo?

Desafio: Crie grupos temáticos com textos sobre esportes, política, tecnologia.

Detecção de Anomalias

Definição: Encontrar textos fora do padrão.

Aplicações:

- Detecção de fraudes
- Comentários ofensivos
- Identificação de spams

Pergunta: Como o “normal” é definido nesse contexto?

Tradução automática: Google Tradutor, DeepL, Whisper

Geração de texto: ChatGPT, Copy.ai, Jasper

Resumo automático: Notícias, artigos acadêmicos, textos legais

Sistemas de diálogo: Assistentes virtuais, SAC, bots em apps

Demonstração – Geração de texto com GPT

```
1 from transformers import pipeline  
2  
3 gerador = pipeline("text-generation", model="gpt2")  
4 resposta = gerador("Era uma vez um robô que", max_length=30)  
5 print(resposta[0]['generated_text'])
```

Observação: Precisa de ambiente com GPU ou Hugging Face Spaces.



Atividade prática – Geração de Texto

Sugestão: Acesse um modelo pré-treinado na HuggingFace ou playgrounds como:

- <https://huggingface.co/spaces>
- <https://platform.openai.com/playground>

Desafio: Crie uma narrativa com base em um tema que você escolher.

Métricas principais:

- Acurácia
- Precisão, Revocação, F1-score
- Perplexidade (em modelos de linguagem)

Demonstre: Como um modelo pode parecer bom, mas errar em minorias.

Exemplo com métricas

```
1 from sklearn.metrics import classification_report  
2  
3 y_true = ['positivo', 'negativo', 'positivo']  
4 y_pred = ['positivo', 'positivo', 'positivo']  
5  
6 print(classification_report(y_true, y_pred))
```

Questões importantes:

- Viés algorítmico (ex: racismo em IA)
- Privacidade dos dados utilizados
- Geração de desinformação
- Responsabilidade e transparência

Discussão: O modelo deve explicar suas decisões?

Concluímos:

- Fundamentos e técnicas de PLN
- Pré-processamento, vetorização e embeddings
- Análise sintática e semântica
- Aprendizado de máquina e aplicações avançadas

Atividade final: A definir