

***INTELIGÊNCIA ARTIFICIAL***

**UA02 / LABORATÓRIO # 5**

**Análise do Código de PLN com Embedding em VectorDB**

**1.) Importação de Bibliotecas**

# Para chunks e embeddings

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from sentence\_transformers import SentenceTransformer

# Para leitura de PDF

import pdfplumber

# Para tratamento de texto

import re

import spacy

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords') # Baixa as stopwords em português

# Para Banco de Dados Vetorial

import chromadb

from langchain\_chroma import Chroma

Esta seção importa todas as bibliotecas necessárias:

- `langchain` para dividir texto em pedaços (chunks)

- `sentence-transformers` para gerar embeddings de texto

- `pdfplumber` para ler arquivos PDF

- `spacy` e `nltk` para processamento de linguagem natural

- `chromadb` para armazenar e consultar vetores

**2.) Função para Ler PDF**

def ler\_pdf(caminho\_pdf):

leitor\_pdf = pdfplumber.open(caminho\_pdf)

texto = ""

for pagina in range(len(leitor\_pdf.pages)):

texto += leitor\_pdf.pages[pagina].extract\_text()

texto = texto.replace("\n", " ")

return texto

Esta função:

\* Abre um arquivo PDF

\* Itera por todas as páginas

\* Extrai o texto de cada página

\* Remove quebras de linha

\* Retorna o texto completo como uma única string

**3.) Leitura do Arquivo PDF**

arquivo\_pdf = "chapeuzinho.pdf"

texto\_pdf = ler\_pdf(arquivo\_pdf)

Lê o conteúdo do arquivo "chapeuzinho.pdf" usando a função anterior.

Isso é como digitalizar o livro inteiro para o computador.

No caso, o texto da Chapeuzinho Vermelho tem 1.741 caracteres.

**4.) Pré-processamento de Texto**

nlp = spacy.load("pt\_core\_news\_sm") # Carrega modelo em português

# Define stopwords (palavras sem significado que serão removidas)

api\_stop\_words = set(stopwords.words('portuguese'))

minhas\_stop\_words = {'a','e','i','o', 'u'}

stop\_words = api\_stop\_words | minhas\_stop\_words # Combina as stopwords

**5.) Função de Tratamento de Texto**

def tratamento\_pln(texto):

# 1.) Normalização - converte para minúsculas

texto = texto.lower()

# 2. Remove números e caracteres especiais

texto = re.sub(r'[^a-zA-Záéíóú\s]', '', texto)

# 3. Tokenização (divide em palavras)

doc = nlp(texto)

# 4. Lematização e remoção de stopwords/pontuação

clean\_tokens = [token.lemma\_ for token in doc

if token.text not in stop\_words and not token.is\_punct]

# 5. Junta os tokens limpos em uma string

clean\_text = ' '.join(clean\_tokens)

return clean\_text

Esta função faz:

1. Normalização para minúsculas

2. Limpeza de caracteres indesejados

3. Divisão em tokens (palavras)

4. Lematização (reduz palavras a sua forma base) e filtragem

5. Reconstrução do texto limpo

**6.) Aplicação do Pré-processamento**

texto\_pdf\_tratado = tratamento\_pln(texto\_pdf)

Aplica o tratamento ao texto extraído do PDF.

Imagine que estamos preparando o texto para uma análise:

- Tudo em letras minúsculas

- Remove pontuação e números

- Tira palavras sem importância ("a", "o", "de", etc.)

- Reduz palavras a sua forma base ("correu" → "correr")

Resultado:

"chapeuzinhar vermelho lobo mau vez menina chamar chapeuzinho vermelho..."

**7.) Divisão em Chunks**

text\_splitter = RecursiveCharacterTextSplitter(chunk\_size=40, chunk\_overlap=10)

chunks = text\_splitter.split\_text(texto\_pdf\_tratado)

Divide o texto em pedaços de 40 caracteres com sobreposição de 10 caracteres.

Dividimos a história em 37 frases pequenas (cada uma com ~40 letras), com pequenas sobreposições.

Exemplos dos pedaços:

\* "chapeuzinhar vermelho lobo mau vez"

\* "mau vez menina chamar chapeuzinho"

\* "vermelho dia pediulhe levar cesta doce"

**8.) Geração de Embeddings**

model = SentenceTransformer('all-MiniLM-L6-v2') # Carrega modelo de embeddings

embeddings = model.encode(chunks) # Gera embeddings para cada chunk

uids = [f"doc\_{i}" for i in range(len(chunks))] # Cria IDs únicos

Este trecho do código: embeddings = model.encode(chunks)

Transforma cada pedaço de texto em vetor numérico (lista de números) que captura seu significado. É como traduzir palavras para uma "língua matemática" que o computador entende.

**9.) Armazenamento no ChromaDB**

client = chromadb.Client() # Cria cliente do banco de dados

collection = client.create\_collection(name="lobomau") # Cria coleção

# Adiciona os chunks, embeddings e IDs ao banco

collection.add(documents=chunks, embeddings=embeddings, ids=uids)

Guardamos todos os pedaços com seus vetores num banco de dados especial (ChromaDB) que entende esses números.

**10.) Consulta ao Banco de Dados**

query\_embedding = model.encode(["vovó é uma comida"]) # Gera embedding da consulta

results = collection.query(query\_embeddings=query\_embedding, n\_results=1) # Busca

Quando você pergunta "vovó é uma comida":

- Transformamos essa pergunta no mesmo tipo de vetor numérico.

- Procuramos no banco o pedaço de texto com vetor mais parecido.

**Exibe resultados:**

for i in range(len(results['ids'][0])):

doc\_id = results['ids'][0][i]

distance = results['distances'][0][i]

document = results['documents'][0][i]

print(f"ID: {doc\_id}")

print(f"Distância: {distance}")

print(f"Documento: {document}")

print("-" \* 40)

O sistema encontrou que o trecho mais relacionado é:

```

"chegar perceber vovó meio diferente"

```

Com similaridade de 0.787 (quanto mais perto de 1, mais similar)

**Saída do Código**

A saída mostra:

\* O texto original do PDF (1741 caracteres)

\* As stopwords usadas (209 palavras)

\* O texto após tratamento

\* Os chunks gerados (37 pedaços)

\* O resultado da consulta "vovó é uma comida":

- Encontrou o chunk mais similar: "chegar perceber vovó meio diferente"

- Com distância (similaridade) de 0.787

**Resumo do Funcionamento**

- Extrai texto de um PDF

- Limpa e normaliza o texto (remove stopwords, pontuação, lematiza)

Divide em pequenos pedaços (chunks)

Converte cada chunk em um vetor numérico (embedding)

Armazena os vetores em um banco de dados vetorial (ChromaDB)

Permite consultas por similaridade (encontra trechos relacionados à consulta)

O código é um pipeline completo de PLN que transforma texto em representações vetoriais e permite buscas semânticas. No exemplo, ao perguntar sobre "vovó é uma comida", o sistema encontrou o trecho que fala sobre a vovó parecer diferente (quando o lobo se disfarçou dela).