

***INTELIGÊNCIA ARTIFICIAL***

**UA09 / LABORATÓRIO # 1**

**ASSUNTO:** IA GENERATIVA - PARTE 1 - PLN E VECTOR DB

**Materiais de Apoio**

**Site oficial do LangChain:**

**https://www.langchain.com/**

**Site oficial do Spacy:**

**<https://spacy.io/>**

**Site oficial do NLTK:**

**<https://www.nltk.org/>**

**Site oficial do ChromaDB:**

**<https://www.trychroma.com/>**

**Instalações para funcionar o Lab:**

pip install langchain

pip install sentence\_transformers

pip install pdfplumber

pip install spacy

python -m spacy download pt\_core\_news\_sm

pip install nltk

pip install chromadb

***Dataset usado:*** *chapeuzinho.pdf*

# PLN com Embedding em VectorDB

# Bibliotecas para PLN

# Para chunks e embeddings

from langchain.text\_splitter import RecursiveCharacterTextSplitter # pip install langchain

from sentence\_transformers import SentenceTransformer # pip install sentence\_transformers

# Para leitura de PDF

import pdfplumber # pip install pdfplumber

# Para tratamento de texto

import re

import spacy # python -m spacy download pt\_core\_news\_sm

import nltk # pip install nltk

from nltk.corpus import stopwords

# Baixando dados do NLTK necessários (se ainda não tiver)

# nltk.download('stopwords') # rodar apenas uma vez

# Bibliotecas para Banco de Dados Vetorial (Vector Database)

import chromadb

from langchain\_chroma import Chroma

# Aplicando PLN - Preparação do texto

# Carregando PDFs

def ler\_pdf(caminho\_pdf):

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

    # page = leitor\_pdf.pages[0]

    texto = ""

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

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

    texto = texto.replace("\n", " ")

    return texto

# Carregar os documentos do PDF

arquivo\_pdf = "C:/IA-Estudos/PLN/PDF/chapeuzinho.pdf"

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

# Tamanho do texto

print("Tamanho do texto em caracteres:",len(texto\_pdf))

# Arquivo PDF original

print(texto\_pdf)

# PLN

# Carregar o modelo de linguagem do spaCy

nlp = spacy.load("pt\_core\_news\_sm")

# Definir stopwords

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

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

stop\_words = api\_stop\_words | minhas\_stop\_words

# Função para fazer o tratamento de linguagem natural usando spaCy

def tratamento\_pln(texto):

    # 1. Normalização: Colocar o texto em minúsculas

    texto = texto.lower()

    # 2. Remoção de números, pontuações e caracteres especiais

    texto = re.sub(r'[^a-zA-Záéíóú\s]', '', texto) # na expressão regular estão as exceções

    # 3. Tokenização com spaCy

    doc = nlp(texto)

    tokens = [token.text for token in doc]

    # 4. Remoção de stopwords, remoção de pontuação

    #    e Lematização (clean\_tokens = tokens lematizados e sem stopwords)

    clean\_tokens = [token.lemma\_ for token in doc if token.text not in stop\_words and not token.is\_punct]

    # 5. Juntar tokens lematizados de volta em uma string

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

    return clean\_text

    #return texto

# Visualizando as Stop Words

print("Tamanho do conjunto stop\_words:",len(stop\_words),"\nStop\_words ordenadas: \n",sorted(list(stop\_words)))

# Chamada de PLN

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

# Tamanho do texto

print("Tamanho do texto em caracteres:",len(texto\_pdf))

# Arquivo PDF tratado

print(texto\_pdf\_tratado)

# Preparação do Texto para BD Vetorial

# Dividindo os documentos em Chunks

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

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

print(chunks, len(chunks))

# Carregar o modelo de Embeddings bem como gerar os Embeddings

model = SentenceTransformer('all-MiniLM-L6-v2')

embeddings = model.encode(chunks)

# Gerando IDs automaticamente

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

## Aplicação do ChromaDB

# Criar o banco de dados

client = chromadb.Client()

#client.delete\_collection("lobomau")

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

#collection = client.get\_collection(name="lobomau")

# Adicionar os documentos ao banco de dados

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

# Realizar a busca usando collection.query

query\_embedding = model.encode(["vovó é uma comida"])

# query\_embedding = model.encode(["vovó é mentirosa"])

# query\_embedding = model.encode(["lobo é mentiroso"])

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

print(results)

# Imprimir os resultados

# Fazendo a varredura sobre os campos 'ids', 'distances' e 'documents'

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)

**Bom Trabalho!!**