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
```

```python
# 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!!