Similaridade de Cossenos

Pedro de Freitas Santos

19 de Maio de 2025

1. Introdução

Este documento representa um projeto que tem o objetivo de desenvolver um código para comparação de dados em um *dataset*, através da fórmula de similaridade de cossenos aprendida em aula na matéria de Álgebra Linear. O *dataset* escolhido para utilização do programa foi o Drugs / medicine description, que contém o nome de 824 medicamentos, suas respectivas descrições sobre seu uso e o link para o site onde os remedios e suas descrições foram coletadas.

2. Descrição Geral do Funcionamento

O código de similaridade de cossenos funciona através da interação no terminal, onde usuário descreve os sintomas que ele esta sentindo e através disso o código compara os seus sintomas com as descrições dos remédios presentes no *dataset*, retornando o remdéio que mais se assemelha com a sua descrição.

3. Tecnologias Utilizadas

As técnologias utilizadas para a criação desse programa foram:

1. Python 3.12.10
2. Ambiente interativo Jupyter Notebook
3. Biblioteca Pandas, Numpy, Sklearn, Nltk e Numpy
4. Excel

4. Código

4.1. Instalando as bibliotecas

Através do Ambiente interativo Jupyter Notebook e utilizando a linguagem de programação Python 3.12.10, o código começa com a instalação das bibliotecas que serão utilizadas para o funcionamento do programa.

```
%pip install nltk scikit-learn pandas

```

Essa linha de comando instala as bibliotecas:

- NLTK: É uma biblioteca de código aberto em Python para Processamento de Linguagem Natural (PLN).
- Scikit-learn: É uma biblioteca de código aberto em Python usada para tarefas de aprendizado de máquina
- Pandas: A biblioteca Pandas é uma ferramenta essencial em Python para manipulação e análise de dados, especialmente dados tabulares (como em planilhas ou tabelas de banco de dados).

4.2. Importando as bibliotecas

Após a instalação para o funcionamento do programa, começaremos a importar as bibliotecas que serão utilizadas e faremos download da lista das stopwords e do módulo punkt_tab.

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
import nltk

nltk.download("punkt_tab")
nltk.download("stopwords")

```

Nessas linhas de código estamos importando todas as bibliotecas e funções que serão utiliazdas no programa, sendo elas:

1. Importa o **pandas** com o apelido pd, facilitando seu chamado durante o código.
2. Importa o **NumPy** com o apelido np, facilitando seu chamado durante o código. O Numpy é uma biblioteca para operações matemáticas e manipulação de arrays.

3. Importa o **TfidfVectorizer** da biblioteca scikit-learn, que transforma textos em vetores numéricos com base no cálculo de TF-IDF (Term Frequency-Inverse Document Frequency). Isso ajuda a quantificar a importância de palavras em um texto.

4. Importa a função **cosine_similarity**, usada para medir a similaridade entre vetores (por exemplo, textos representados por TF-IDF) com base no cosseno do ângulo entre eles. Um valor mais próximo de 1 indica maior similaridade.

5. Importa a lista de **stopwords** (palavras comuns como "a", "o", "de", etc.) da biblioteca nltk, que são normalmente removidas durante o pré-processamento de texto.

6. Importa o **word_tokenize**, uma função da nltk para dividir um texto em palavras (tokenização).

7. Importa o módulo **string**, que contém funções e constantes úteis para lidar com texto, como string.punctuation, que é uma string com todos os sinais de pontuação.

8. Importa a biblioteca **nltk**, usada para tarefas de processamento de linguagem natural (PNL).

9. Faz o download de **punkt_tab**, um modelo de tokenização necessário para o word_tokenize funcionar.

10. Faz o download da lista de **stopwords** (palavras irrelevantes) da nltk.

4.3. Lendo o Dataset

```
df_remedios = pd.read_csv("drugs.csv")

```

1. **df_remedios** É a variável que recebe o DataFrame resultante da leitura do CSV. Essa variável agora contém todos os dados da tabela do arquivo **drugs.csv**

2. **pd.read_csv** é uma função do pandas é usada para ler arquivos CSV e transformá-los em um DataFrame, que é uma estrutura de dados em forma de tabela.

```
df_remedios.head()

```

A função **.head()**: é um método do pandas que retorna as primeiras linhas do DataFrame, por padrão as 5 primeiras.

4.4. Função para tratamento dos dados

```
def tratando_dados(texto):
    texto = str(texto).lower()
    texto.translate(str.maketrans("", "", string.punctuation))
    tokens = word_tokenize(texto)
    stop_words = set(stopwords.words("english"))
    tokens = [palavra for palavra in tokens if palavra not in stop_words]
    tratamento_dados = " ".join(tokens)

    return tratamento_dados

```

A função **tratando_dados** serve para limpar e preparar os dados brutos do *dataframe* para a análise. A função converte tudo para string e transforma todas as letras em minúsculas, após isso, remove toda pontuação do texto(como virgulas, pontos, exclamações, etc). Utilizando a função **tokens = word_tokenize(texto)** ele divide o texto em palavras e usando a função **stop_words = set(stopwords.words("english"))** em junção a **tokens = [palavra for palavra in tokens if palavra not in stop_words]** ele pega uma lista de palavras comuns (stopwords) em inglês e remove dos tokens todas as palavras que forem stopwords, ou seja, deixa apenas as palavras consideradas mais relevantes. Por fim, usa a função **tratamento_dados = " ".join(tokens)** para junta os tokens de volta em uma única string, separando por espaço, retornando o texto final limpo e tratado, pronto para análise.

4.5. Função para recomendar os remédios

```
df_remedios["descri  o"] = df_remedios["description"].apply(
    tratando_dados)

```

Cria uma nova coluna com o nome description com os dados já tratados

```
def recomenda_remedio(recomenda, df_remedios):
    remedios_tratados = tratando_dados(recomenda)
    tfidf = TfidfVectorizer()
    descricoes = df_remedios["descri  o"].tolist() + [remedios_tratados]
    tfidf_matriz = tfidf.fit_transform(descricoes)
    sim_cosseno = cosine_similarity(tfidf_matriz[-1], tfidf_matriz[:-1])
    remedio_recomendado = np.argmax(sim_cosseno)

    return df_remedios["drugName"].iloc[remedio_recomendado]

```

função **recomenda_remedio** tem como objetivo recomendar o remédio mais apropriado com base em uma descrição fornecida pelo usuário. A descrição do usuário é tratada pela função **tratando_dados**, depois, a função utiliza o **TfidfVectorizer**, uma ferramenta da biblioteca scikit-learn, para converter os textos (descrições dos remédios e a descrição do usuário) em vetores numéricos com base na importância das palavras. Isso permite quantificar os textos de

forma que seja possível compará-los matematicamente. A entrada do usuário é adicionada à lista de descrições dos remédios. Em seguida, o **fit_transform** do TfidfVectorizer transforma todas essas descrições em vetores. Com esses vetores, a função calcula a similaridade do cosseno entre a entrada do usuário e todas as descrições do DataFrame. A função então identifica qual descrição de remédio tem a maior similaridade com o texto do usuário e retorna o nome (drugName) do remédio correspondente.

```
consulta = input("Digite os seus sintomas em ingl s:")
recomendado = recomenda\_remedio(consulta, df_remedios)
print("O rem dio recomendado   :", recomendado)

```

faz a interação final com o usuário e exibe a recomendação de remédio com base nos sintomas digitados.

5 Conclusão

Neste projeto, foi possível desenvolver um sistema básico de recomendação de remédios utilizando técnicas de PLN, a partir da entrada textual do usuário. Esse processo demonstra como é possível aplicar métodos para ajudar na tomada de decisões, auxiliando o usuário a encontrar possíveis medicamentos com base em suas necessidades descritas.