CENTRAL UNIVERSITY OF ECUADOR

SI3-002

ENGINEER:

WILLIAM FERNANDO VASQUEZ ITURRALDE

NAME:

Adrian Calero

TOPIC:

Avance del Proyecto()

DATE:

22/11/2023

Software de Base de Datos:

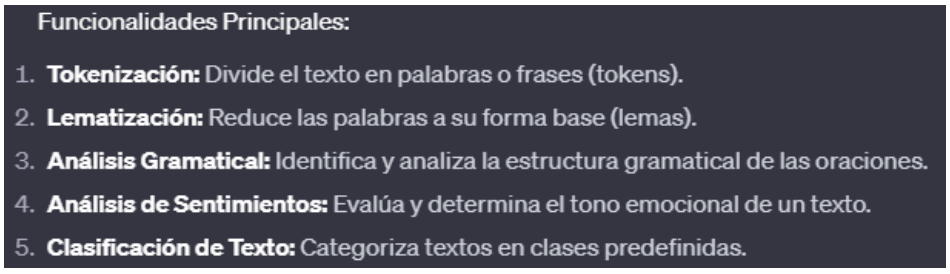
PostgreSQL. Puedes interactuar con estas bases de datos utilizando bibliotecas como mysql-connector o psycopg2 en Python.

Base de Datos:

HealthData.gov

Publicador: Gobierno de EE. UU

NLTK (Natural Language Toolkit) son bibliotecas de Python que se utilizan comúnmente en tareas relacionadas con el procesamiento de lenguaje natural (NLP) y aprendizaje automático, respectivamente.



scikit-learn es una biblioteca de aprendizaje automático que proporciona herramientas sencillas y eficientes para análisis predictivo y minería de datos. Se utiliza para tareas de clasificación, regresión, clustering, reducción de dimensionalidad, selección de modelos y preprocesamiento de datos.

1. Instalar las siguientes bibliotecas:

```bash
pip install nltk
pip install scikit-learn
```

2. Instalar librerías y descargar recursos:

```python
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import string
import random

nltk.download('punkt')
nltk.download('stopwords')
```

3. Bajar una base de datos

   base de datos de enfermedades y síntomas en línea. Asegúrate de que la base de datos esté en un formato que puedas procesar fácilmente en Python, como un archivo CSV.

4. Procesar datos:

```
from nltk.corpus import stopwords

def preprocess_text(text):
    # Convertir a minúsculas
    text = text.lower()
    # Eliminar puntuación
    text = text.translate(str.maketrans("", "", string.punctuation))
    # Eliminar palabras comunes (stop words)
    stop_words = set(stopwords.words('english'))
    tokens = nltk.word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

5. Crear modelador TF-IDF

```
python

def create_tfidf_model(corpus):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(corpus)
    return tfidf_matrix
```

6. Lógica del chatbot

```
def get_response(user_input, corpus, tfidf_matrix):
    user_input = preprocess_text(user_input)
    user_vector = vectorizer.transform([user_input])

    # Calcular similitud del coseno entre la entrada del usuario y e
    similarity_scores = cosine_similarity(user_vector, tfidf_matrix)

    # Obtener el índice de la respuesta más relevante
    max_index = similarity_scores.argmax()

    return corpus[max_index]
```

7. Interacción

```
def chat():
    # Reemplace 'corpus' con la lista de respuestas posibles
    corpus = ["Enfermedad A", "Enfermedad B", "Enfermedad C"]
    tfidf_matrix = create_tfidf_model(corpus)

    print("¡Hola! Soy un chatbot de medicina. ¿En qué puedo ayudarte?")
    while True:
        user_input = input("Usuario: ")
        if user_input.lower() == 'adios':
            print("¡Hasta luego!")
            break

        response = get_response(user_input, corpus, tfidf_matrix)
        print("Chatbot: {}".format(response))
```

TF-IDF (Term Frequency-Inverse Document Frequency) es una técnica utilizada en procesamiento de lenguaje natural y recuperación de información para evaluar la importancia relativa de una palabra en un documento con respecto a una colección de documentos.

En el contexto del ejemplo del chatbot de medicina, el modelo TF-IDF se utiliza para convertir la información de texto (síntomas de enfermedades) en una representación numérica que puede ser utilizada para calcular la similitud entre la entrada del usuario y las enfermedades en la base de datos.