Universidad Tecnológica de Honduras

Propuesta de desarrollo de:

ChatbotCM - Chatbot conversacional sobre consultas medicas

Asignatura:

Inteligencia Artificial (IAE-0611)

Catedrático:

Ing. Wilson Octaviano Villanueva Castillo

Presentado por:

Elsy Paola Amaya Lazo - 201910040094

Orlin Fabricio Lagos - 201710110523

Sábado, 15 de abril de 2023

Índice

Introducción

Este proyecto es una aplicación web desarrollada en Flask, que ofrece un chatbot diseñado para proporcionar información médica básica. Además, incluye una sección de preguntas y respuestas donde los usuarios pueden buscar respuestas a preguntas específicas, agregar nuevas preguntas, deshabilitar preguntas antiguas, editar preguntas existentes, y una sección de gestión de usuarios donde los administradores pueden agregar nuevos usuarios, deshabilitar usuarios existentes y editar sus roles. La aplicación utiliza una base de datos SQLite3 para almacenar la información del chat y la información de los usuarios. También utiliza la biblioteca de reconocimiento de voz y la biblioteca de procesamiento del lenguaje natural (NLTK) para mejorar la funcionalidad del chatbot.

El archivo "FnNLTK.py" utiliza la biblioteca Natural Language Toolkit (NLTK) para crear un chatbot en español. Se conecta a una base de datos SQLite para obtener preguntas y respuestas para entrenar al chatbot. Utiliza técnicas de preprocesamiento de texto como la tokenización, eliminación de palabras irrelevantes y lematización para mejorar la calidad del entrenamiento. También utiliza la biblioteca Scikit-learn para la vectorización de preguntas y la medición de similitud coseno para seleccionar la respuesta más adecuada para una pregunta dada.

Por otro lado, el archivo "FnSpeechRecognition.py" utiliza la biblioteca SpeechRecognition para convertir audio de voz en texto en español. Utiliza la clase Recognizer para detectar el audio y luego lo transcribe a texto utilizando el servicio de reconocimiento de voz de Google. Si el audio no se detecta o no se puede transcribir, devuelve un mensaje de error.

Arquitectura del sistema

Este es un proyecto de Flask que consta de varias rutas y funciones, que se utilizan para crear una aplicación web de chatbot de Inteligencia Artificial con funcionalidades de administración de usuarios, preguntas y errores.

La arquitectura del proyecto es la siguiente:

1. Primero, se importan las bibliotecas necesarias: Flask, render_template, request, redirect, url_for, session, flash, FnSQLite3, Speed_to_text y Chat de NLTK.
2. Luego se crea una instancia de la clase Flask con el nombre de la aplicación y se establecen la carpeta de plantillas y la carpeta estática.
3. Se define una función de ayuda llamada check_login(), que verifica si un usuario ha iniciado sesión.
4. Se define una ruta '/login' y una función login() que muestra la página de inicio de sesión.
5. Se define una ruta '/login' con el método POST y una función login_post() que procesa el inicio de sesión y verifica los datos del usuario.
6. Se define una ruta '/logout' y una función logout() que elimina la información del usuario de la sesión.
7. Se define una ruta '/home' y una función home() que muestra las preguntas en la base de datos.
8. Se define una ruta '/question' con el método POST y una función save_question() que guarda una nueva pregunta en la base de datos.
9. Se define una ruta '/question/delete/<int:id>' con el método POST y una función delete_question(id) que elimina una pregunta de la base de datos.

10. Se define una ruta '/edit_question_modal' con el método POST y una función edit_question_modal() que actualiza una pregunta en la base de datos.
11. Se define una ruta '/error' y una función error() que muestra los errores en la base de datos.
12. Se define una ruta '/error/delete/<int:id>' con el método POST y una función delete_error(id) que elimina un error de la base de datos.
13. Se define una ruta '/user' y una función user() que muestra los usuarios en la base de datos.
14. Se define una ruta '/save-user' con el método POST y una función save_user() que guarda un nuevo usuario en la base de datos.
15. Se define una ruta '/user/delete/<int:id>' con el método POST y una función delete_user(id) que elimina un usuario de la base de datos.
16. Se define una ruta '/edit_user_modal' con el método POST y una función edit_user_modal() que actualiza un usuario en la base de datos.

En resumen, este proyecto es un chatbot con una funcionalidad de administración de usuarios, preguntas y errores.

Tecnologías y herramientas utilizadas

Este es un proyecto desarrollado en Flask, un framework de Python utilizado para el desarrollo de aplicaciones web. Las tecnologías y herramientas utilizadas en este proyecto son las siguientes:

1. **Flask:** framework de Python para desarrollo web
2. **Jinja2:** motor de plantillas utilizado para renderizar las páginas web
3. **SQLite3:** sistema de gestión de bases de datos utilizado para almacenar preguntas, errores y usuarios.

4. **Fnpy:** librería personalizada para el procesamiento del lenguaje natural.
5. **FnSpeechRecognition:** librería personalizada para la transcripción de voz a texto
6. **FnNLTK:** librería personalizada para procesar el lenguaje natural.
7. **HTML:** lenguaje de marcado utilizado para estructurar el contenido de las páginas web.
8. **CSS**: lenguaje de hojas de estilo utilizado para dar estilo a las páginas web.
9. **JavaScript:** lenguaje de programación utilizado para añadir interactividad a las páginas web.
10. **Bootstrap:** framework de CSS utilizado para crear diseños web responsivos y móviles.

Estructura de la base de datos

La estructura de la base de datos de este proyecto es una base de datos SQLite. La base de datos contiene cuatro tablas: user, question y error. La tabla user almacena la información del usuario y contiene los campos id, user_name, password, rol y state. La tabla question almacena las preguntas y respuestas del chatbot y contiene los campos id, question, answer y state. La tabla error almacena las preguntas mal interpretadas por el chatbot y contiene los campos id, question y state. El archivo FnSQLite3.py define funciones para acceder a la base de datos y realizar operaciones como iniciar sesión, buscar preguntas, agregar preguntas, editar preguntas, eliminar preguntas, buscar errores, agregar errores, eliminar errores, buscar usuarios, agregar usuarios, editar usuarios y eliminar usuarios.

Funcionalidades implementadas

Este es un proyecto de chatbot médico, por lo que las funcionalidades implementadas están relacionadas con las tareas que puede realizar el chatbot. Algunas de las funcionalidades implementadas son:

- Inicio de sesión: el chatbot verifica si el usuario está autenticado para permitir el acceso a las funciones protegidas por contraseña.
- Búsqueda de preguntas: el chatbot puede buscar preguntas en una base de datos de preguntas y respuestas médicas.
- Creación de nuevas preguntas: los usuarios pueden agregar nuevas preguntas y respuestas médicas a la base de datos.
- Edición de preguntas: los usuarios pueden editar preguntas y respuestas médicas existentes en la base de datos.
- Eliminación de preguntas: los usuarios pueden eliminar preguntas y respuestas médicas existentes en la base de datos.
- Registro de errores: los errores que se produzcan durante la ejecución del chatbot se registran en una base de datos para su posterior revisión.
- Gestión de usuarios: los usuarios pueden ser agregados, editados o eliminados.
- Interacción de chat: el chatbot es capaz de proporcionar información médica básica en función de las preguntas formuladas por el usuario.
- Reconocimiento de voz: se ha implementado una funcionalidad de reconocimiento de voz para que el chatbot pueda recibir preguntas habladas por el usuario.

Conclusión

Consiste en la creación de un chatbot que utiliza un modelo de aprendizaje automático para responder preguntas y una herramienta de reconocimiento de voz para permitir a los usuarios interactuar con el chatbot mediante comandos de voz. El archivo FnNLTK.py contiene el código para entrenar el modelo de chatbot utilizando el paquete Natural Language Toolkit (NLTK) y la biblioteca scikit-learn para crear un modelo de vectorización de términos de frecuencia de documentos invertidos (TF-IDF) y el cálculo de similitud coseno para encontrar la respuesta más adecuada a la pregunta del usuario. El archivo FnSpeechRecognition.py utiliza la biblioteca SpeechRecognition para capturar y transcribir los comandos de voz del usuario. En general, el proyecto tiene como objetivo facilitar la interacción con el chatbot a través de una herramienta de reconocimiento de voz y proporcionar respuestas precisas y relevantes utilizando un modelo de aprendizaje automático entrenado previamente.