FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL

INFORME DE AVANCES

TEMA:

Chatbot que responda a preguntas frecuentes utilizando un conjunto de documentos institucionales.

GRUPO DE TRABAJO:

- *Edgar David Martínez*
- *Jandri Giovanni Villavicencio*

MANUAL DE USUARIO

REPOSITORIO EN GITHUB

- *Link: https://github.com/Edmartinez28/Chatboot-Olaf*

DEPENDENCIAS

- **Librerías**
- PyPDF2
- Spacy
- Pandas
- Numpy
- Openai
- Sklearn
- NLTK
- Ast

LIBRERÍAS

En el desarrollo de Olaf, nuestro objetivo principal ha sido crear un chatbot inteligente y versátil que ofrezca respuestas contextuales y precisas a las consultas de los usuarios. Para lograr este propósito, hemos seleccionado cuidadosamente algunas de las mejores librerías disponibles en el ámbito del procesamiento del lenguaje natural y la manipulación de documentos. Entre las principales dependencias que han potenciado las capacidades de Olaf se encuentran PyPDF2, Spacy y NLTK.

- PyPDF2
    - Para el manejo eficiente de documentos en formato PDF dentro de Olaf, se ha implementado la librería PyPDF2. Esta elección se fundamenta en su capacidad para extraer texto de archivos PDF de manera sencilla y rápida. PyPDF2 ha demostrado ser esencial para la comprensión de información contenida en documentos PDF, permitiendo que Olaf procese y responda de manera efectiva a consultas basadas en este tipo de archivos. Los beneficios de esta librería incluyen su facilidad de uso y su

capacidad para integrarse de manera objetiva en el flujo de trabajo del chatbot, mejorando así la versatilidad de Olaf al interactuar con documentos PDF.

- Spacy
    - La inclusión de la librería Spacy en el desarrollo de Olaf ha sido esencial para llevar a cabo tareas avanzadas de procesamiento del lenguaje natural (NLP). Spacy proporciona herramientas robustas para el análisis sintáctico, reconocimiento de entidades y desambiguación léxica. Esto ha mejorado significativamente la capacidad de Olaf para comprender y responder de manera más contextual y precisa a las consultas de los usuarios. Además, la eficiencia computacional de Spacy contribuye a un rendimiento óptimo, lo que es crucial para la respuesta en tiempo real característica de los chatbots.
- NLTK
    - El uso de NLTK en el desarrollo de Olaf ha sido clave para una amplia variedad de tareas relacionadas con el procesamiento del lenguaje natural. NLTK ofrece recursos esenciales como corpus lingüísticos y algoritmos para el análisis de texto. Se ha integrado en Olaf para tareas como tokenización, análisis de sentimientos y procesamiento de texto enriquecido. La robustez y versatilidad de NLTK han permitido que Olaf comprenda y responda a la diversidad de expresiones lingüísticas de los usuarios, mejorando así la calidad y relevancia de las respuestas generadas.

Una vez mencionadas algunas de las librerías utilizadas en el desarrollo de Olaf, podemos afirmar que la selección de PyPDF2, Spacy, NLTK, y otras librerías para el desarrollo de Olaf se ha fundamentado en las capacidades específicas que cada librería aporta al contexto de un chatbot. Estas decisiones han fortalecido las capacidades de Olaf en términos de comprensión del lenguaje natural, procesamiento de documentos PDF y análisis contextual, contribuyendo así a una experiencia de usuario más enriquecedora y eficiente.

Desarrollo de la interfaz de Olaf utilizando Django

La elección de Django para el desarrollo de la interfaz de Olaf se ha basado en la necesidad de crear una experiencia de usuario excepcional y enriquecedora. Las ventajas específicas de Django se han traducido directamente en mejoras tangibles en la interfaz de Olaf, optimizando la presentación visual y la interactividad. Aquí destacamos cómo Django ha contribuido al desarrollo específico de Olaf:

- Imágenes en Forma de Avatares:
    - Django nos facilitó la gestión y presentación de imágenes de manera eficiente. Hemos implementado esta funcionalidad para incorporar avatares que representan tanto al usuario como a Olaf. Estos avatares no solo agregan un toque personalizado a la conversación, sino que también ayudan a los usuarios a identificar fácilmente quién está generando cada mensaje.
- Contenedores de Texto Responsivos:
    - La capacidad que tiene Django para trabajar con plantillas dinámicas nos ha permitido la creación de contenedores de texto responsivos. Enfocándonos en la comodidad del usuario esto nos asegura que los mensajes se presenten de manera clara y legible, adaptándose de manera efectiva a diferentes tamaños de pantalla y dispositivos.
- Alineación de Contenedores y Separación por Color:
    - Django ofrece varias herramientas que simplifican la personalización del diseño. Hemos aprovechado estas capacidades para alinear y separar visualmente los contenedores de mensajes. Los mensajes del usuario y de Olaf se presentan en

contenedores diferenciados por colores, mejorando la legibilidad y la comprensión visual de la conversación.

- Alineación Dinámica Según el Remitente del Mensaje:
    - Una de las características clave implementadas es la alineación dinámica de los mensajes según el remitente. Los mensajes del usuario y de Olaf se presentan con alineaciones distintas, lo que brinda una representación clara de quién está participando en la conversación. Django facilitó esta personalización, permitiendo una adaptación intuitiva del diseño según el origen del mensaje.

En si podemos decir que la elección de Django no solo se basó en sus ventajas generales como framework, sino también en su capacidad para abordar específicamente las necesidades de Olaf. La incorporación de avatares, contenedores de texto responsivos, alineación cuidadosa y diferenciación visual por remitente ha transformado la interfaz de Olaf, proporcionando una experiencia de usuario fluida y estéticamente agradable.

PROCEDIMIENTO PARA REPLICAR CHATBANKER OLAF

Lo que se usó para poder hacer la implementación de Olaf fue la creación de un virtualenv en python haciendo uso de la versión 3.8.4 de Python, dentro del entorno virtual se encontraban instalados todas las librerías pertenecientes al archivo que se adjunta de requirements.txt, en esta se describen cada una de las dependencias que se implementaron para poder hacer uso de este proyecto.

PASOS PARA REPLICAR EL PROYECTO EN LOCAL

1. Instalar Git
2. Instalar Python y pip
3. Clonar el repositorio

Abre tu terminal y navega al directorio donde deseas clonar el repositorio. Luego ejecuta el siguiente comando:

git clone https://github.com/Edmartinez28/Chatboot-Olaf

4. Crear un entorno virtual (opcional pero recomendado)

Se recomienda usar un entorno virtual para poder ejecutar el proyecto sin problemas

python -m venv venv

Luego, activa el entorno virtual:

- En Windows:

.\venv\Scripts\activate

- En Unix o MacOS:

source venv/bin/activate

5. Instalar dependencias

Dentro del directorio del proyecto, utiliza pip para instalar las dependencias del proyecto, las cuales se encuentran adjuntas al proyecto en el repositorio de github

pip install -r requirements.txt

6. Aplicar migraciones

Para poder realizar la ejecución del ciclo se necesitan aplicar las migraciones que ya vienen en el proyecto, para ello se ejecuta el comando

python manage.py migrate

7. Iniciar el servidor de desarrollo

Finalmente, iniciamos el servidor de desarrollo de Django con el siguiente comando

python manage.py runserver

Con ello debe de correr el proyecto en local en http://localhost:8000 en su navegador para ver la aplicación en ejecución.