**Documentación Asistente Banco Galicia**

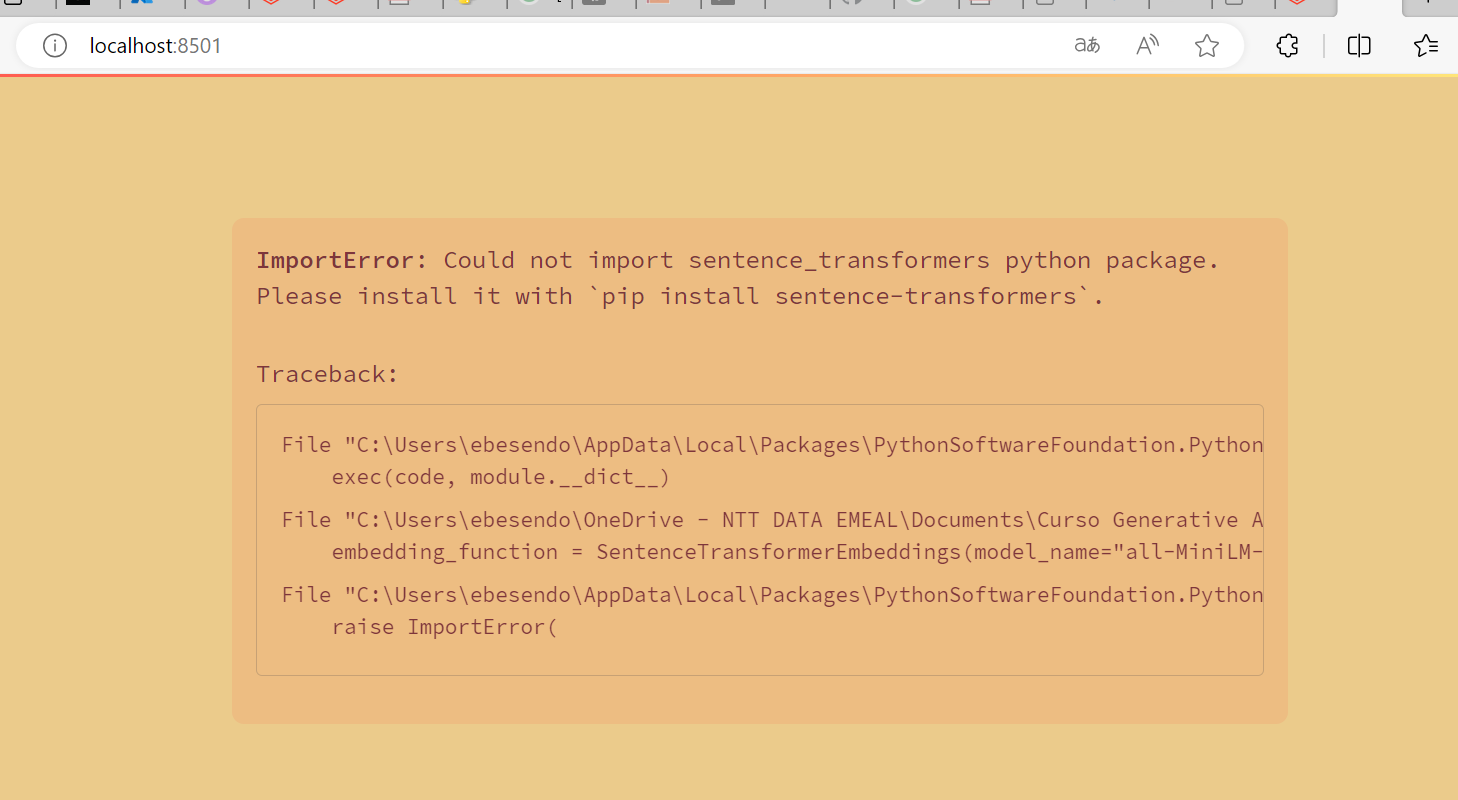
El equipo está formado por Juan Pablo Carrizo, María De La Paz Casaux y Esteban Ariel Besendo.

El objetivo del taller era diseñar y construir un asistente virtual local utilizando LangChain, con memoria integrada, personalidad, y utilizando técnicas avanzadas de Prompting.

A lo largo del trabajo nos encontramos con diversas dificultades.

La primera complicación fue la de crear el entorno virtual. Se instalaron varias librerías y módulos y a veces aparecían errores que nos pedían librerías que ya estaban instaladas.

Otro fallo parecido fue el de la VPN. A veces al abrir VSC y ejecutar el código nos mostraba el siguiente error:



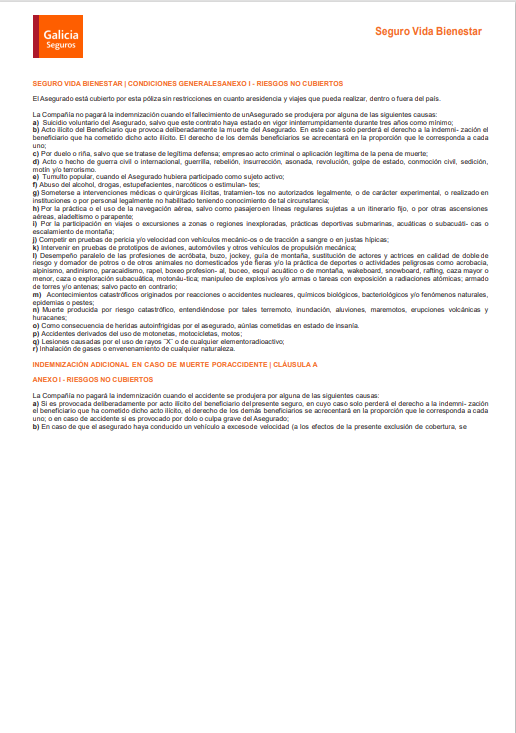
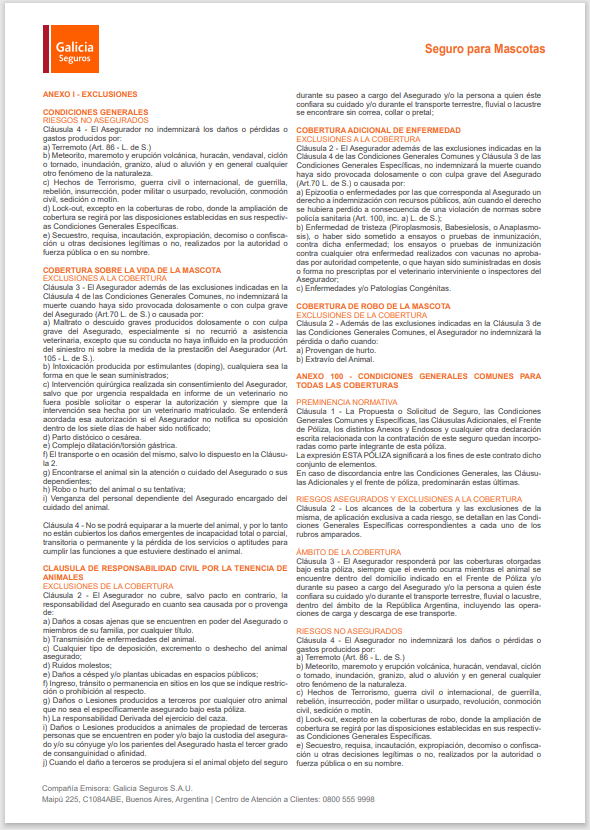
Captura de pantalla de computadora

Descripción generada automáticamente

Este error desaparecía al conectarse a la vpn y volver a ejecutar.

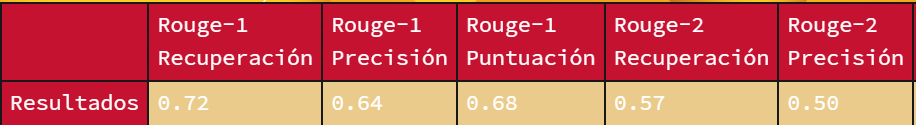
También tuvimos problemas con los documentos. En un principio utilizamos pdf de dos columnas y el asistente no los reconocía bien por lo que los transformamos hasta llegar a un pdf de una columna.

Transformación de PDF:



Otra dificultad con la que nos encontramos fue la de investigar la librería Streamlit para cambiar su interfaz en base a nuestras necesidades. Se cambió el fondo, logo, se introdujeron avatares en los diálogos y se modificó el set bar, por último, si incorporó un dataframe como salida para las métricas.

Salida con dataframe:



Asistente final con interfaz modificada:



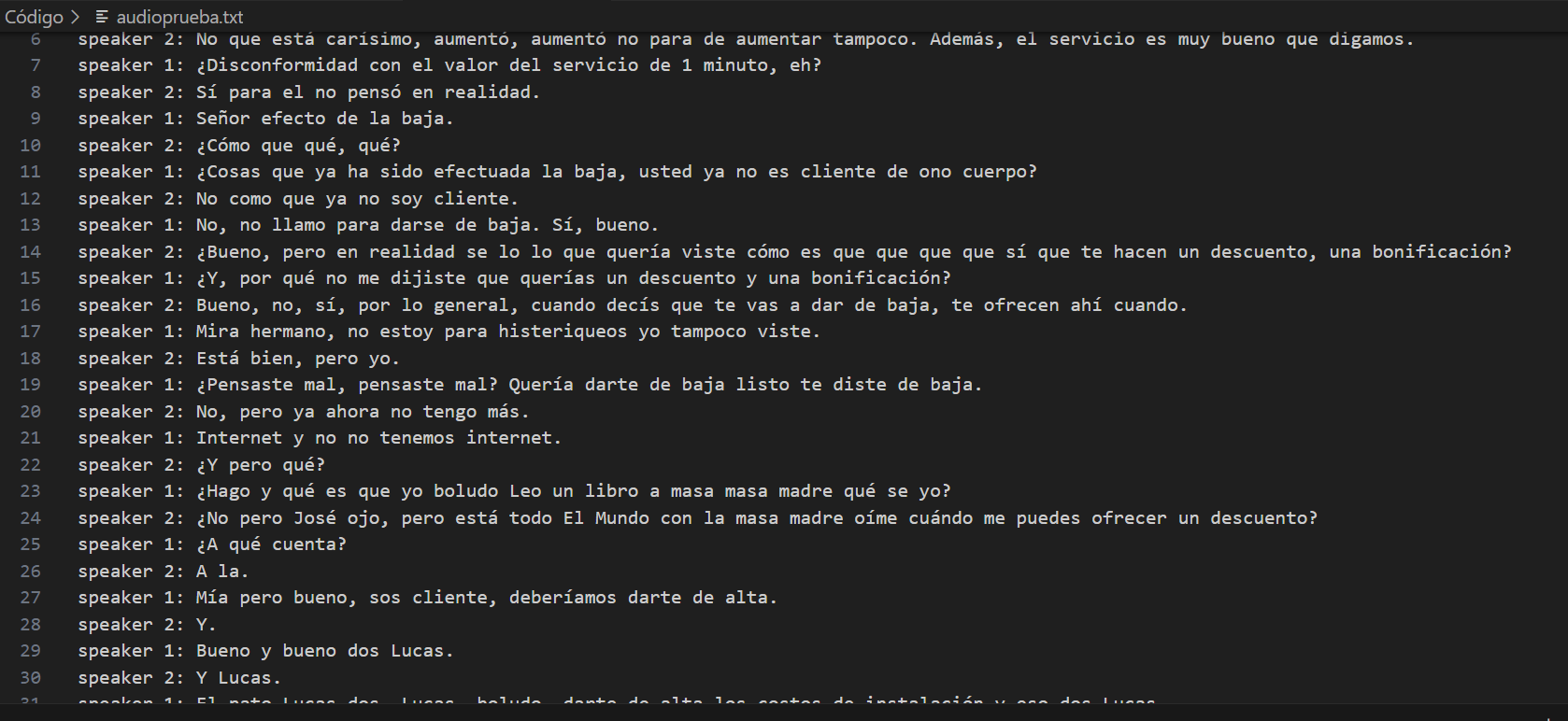
La memoria también nos trajo problemas. No lográbamos que funcionara. Al consultar con Nacha nos indicó que el problema estaba en que ésta estaba en un único archivo junto al resto del código y que se reiniciaba al hacer una pregunta por lo que separamos las funciones a un archivo “functions.py” y en un archivo “main.py” quedaron los archivos de streamlit.

Cambiamos la función para calcular embeddings, pasamos de SentenceTransformerEmbeddings a OpenAIEmbeddings.

También separamos en functions.py la memoria, el template, el prompt y la cadena fuera de la función conversation\_complete.

Decidimos tener en cuenta el consejo de Nacha y probamos con un PDF que no estaba relacionado a Galicia Seguros y seguíamos con los errores.

Al seguir con problemas en la lectura del documento y ver que las respuestas que obteníamos del asistente eran inconsistentes, optamos por cambiar el código y comenzar a trabajar con un documento de texto que obtuvimos de la transcripción de una llamada (El método de transcripción proviene del proyecto Galicia Seguros Speech to Text – Análisis de Sentimiento, aunque la llamada que se usó es una llamada ficticia descargada de internet ya que el método requiere servicios pagos como “diarización”).

Archivo .txt utilizado:

En este punto la mejora en el asistente fue notoria y pudimos seguir con la integración de métricas.

Las métricas utilizadas fueron las siguientes: Rouge-1, Rouge-2, y Rouge-L (puntuación, recuperación y precisión) y Tasa de error de conversación (Cer).

Con estas métricas buscamos ver la efectividad del asistente en cuanto al porcentaje de palabras del resumen generado que se encuentra en el resumen de referencia.

Con Rouge-1 vemos las palabras que se asemejan, con Rouge-2 la semejanza entre bigramas (pares de palabras) y ya con Rouge-L y a modo más general vemos el tema de la coherencia.

Al momento de generar este documento volvimos a realizar la prueba con el asistente anterior en la que usábamos un documento distinto y esta vez funcionó casi perfecto (falló un poco la memoria).

A modo de conclusión, entendimos la importancia de la creación del asistente con sus respectivos elementos. Cabe destacar que con el asistente tenemos una base para mejorarlo y adaptarlo de acuerdo con las necesidades de un proyecto. Recordemos que en el proyecto las transcripciones de las llamadas se obtienen a través de servicios de Azure que son pagos, por ende, sería un tema por tratar con el cliente en base a su presupuesto (a más llamadas transcriptas, mayores gastos). En cuanto al asistente, quedaría adaptar el código para recibir más llamadas.

Asistente en acción:



Instrucciones de instalación:

Descargar los archivos (“requeriments”, “sql\_app\_functions” ,“sql\_app” , “config”, “aclaraciones” , “audioprueba”, “Logo\_galicia” , “Fondo\_galicia”) y colocarlos en una misma carpeta. En esa misma carpeta crear el archivo .env desde la terminal de la siguiente manera:

python -m venv env

env\scripts\activate

Luego en el archivo .env colocar lo siguiente:

OPENAI\_API\_TYPE=azure

OPENAI\_API\_VERSION=2023-07-01-preview

OPENAI\_API\_BASE=https://nttazuredemoopenai.openai.azure.com/

OPENAI\_API\_KEY=reemplazar este texto por una clave válida

En el archivo “sql\_app.py” reemplazar las rutas correspondientes a los archivos de logo y de fondo (“set\_logo” y “set\_background” en la función “setup”).

Para ejecutar: (directorio donde están los archivos descargados)> streamlit run sql\_app.py