TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL (TUIA)

PROCESAMIENTO DE LENGUAJE NATURAL (PLN)

Docentes de la cátedra:

Prof. Juan Pablo Manson

Prof. Alan Geary

Trabajo Final

Estudiante:

Bruno Longo - L-3162/3

AÑO 2024 (1er CUATRIMESTRE), Rosario.

INTRODUCCIÓN:

El presente informe sobre el trabajo final de la materia consta de la explicación de un modelo de lenguaje natural realizado como tarea para el fin de la materia. El modelo debía ser RAG (Retrieval Augmentation generator) implementado en un chatbot, el cual debía cumplir el rol de un experto en cierta temática a elección.

Puesto que soy psicólogo de formación, mi intención fue implementar un chatbot que utilizara información referente a psicoanálisis.

Paso a detallar:

DESARROLLO:

El Notebook de Jupyter consta de la instalación de librerías y 5 segmentos relevantes:

1. **Base de datos vectorial:**

Se descargan 3 libros de psicoanálisis relevantes para teoría (“Diccionario de Psicoanálisis”, “Seminario 20”, “Estructuras Clínicas y Psicoanálisis”) para alimentar la base de datos y almacenarlos en formato PDF.

Luego se procede al Split de las cadenas de texto, con LangChain y su RecursiveTextSplitter (chunks de 750 y 250 de overlap, valores implementados durante el uso, relativamente grandes por ser contenido conceptual). Esto es almacenado en diccionarios, uno por libro.

Posteriormente se realizan los embeddings pertinentes mediante ChromaDB y su almacenamiento como colecciones.

Se define la funcion BusquedaSemantica para devolver los resultados mas cercanos ante una query, mediante la distancia de coseno.

2. **Base de datos tabular:**

Se crea manualmente una tabla compuesta por Título, Autor, Tipo y Año para cargar información sobre las publicaciones más relevantes de algunos autores de psicoanálisis. luego se crea un archivo .csv y un dataframe de pandas para ser utilizado por la búsqueda.

Se crea la función BusquedaTabular que dispone de una búsqueda Fuzzy para ser flexible respecto a los nombres, con un umbral de similitud de 70/100.

3. **Base de datos de grafos:**

Son creadas la clase y métodos dispuestos por la documentación de Wikidata para la lectura, descarga y muestra (en forma de listas) de las querys y sus resultados en SPARQL.

Luego son cargadas las querys referentes a 5 autores fundamentales de psicoanálisis (Freud, Lacan, Klein, Winnicott y Jung) que dispondrán la búsqueda de sus datos biográficos.

4. Clasificación y Funcionamiento del Chatbot:

En esta etapa se determinan las clases y funciones para el funcionamiento del chatbot, desde las plantillas de recepción y emisión de prompts y clasificación de los mismos.

Mediando las función ConexionLLM se crea la interfaz con el modelo Zephyr-7b-beta que será el encargado de interpretar tanto las preguntas como el contexto, para generar las respuestas pertinentes.

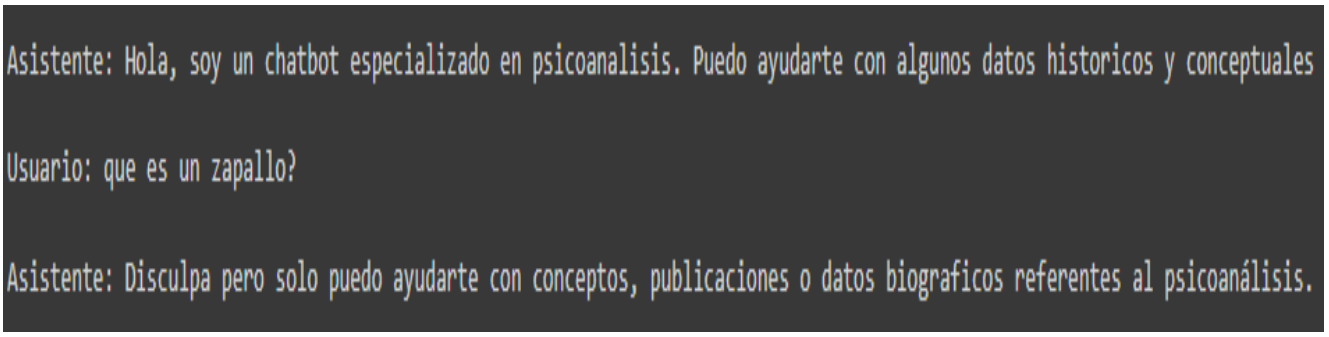
En primera instancia se limpian los prompts, para luego utilizar el modelo a fin de obtener la clasificación de las categorías. En este trabajo se consideraron 4 categorías referentes al psicoanálisis:

- **Teoría y conceptos.**
- **Publicaciones.**
- **Datos Biográficos.**
- **Fuera de las 3 categorías anteriores.**

Es el mismo modelo nombrado anteriormente el que se encarga de clasificar la pregunta en alguna de estas 4 categorías, para ello la función ClasificadorBaseDeDatos reconocerá mediante instrucciones semánticas y retornará el número de la categoría y un porcentaje de similitud para cada una de ellas, a fin de ubicar la base de datos (definida de 1 a 3) más relevante.

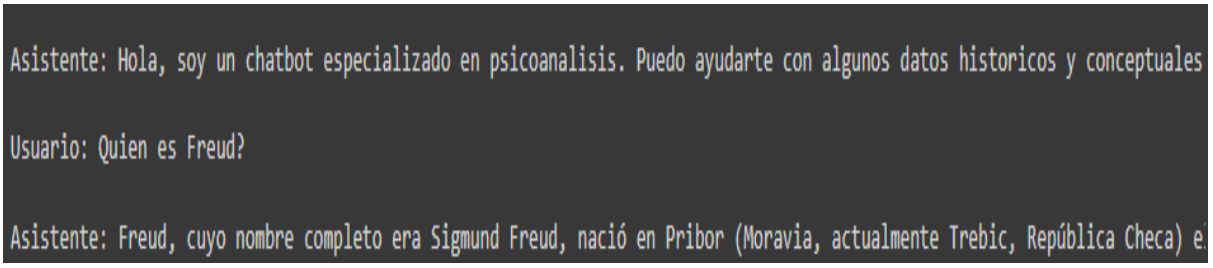
Luego, en el funcionamiento propio del chatbot, se utilizará también el modelo "MoritzLaurer/mDeBERTSa-v3-mnli-xnli" para clasificación zero-shot, para recategorizar ya no el tipo de base de datos, si no el material más adecuado dentro del disponible para contestar a la pregunta.

En caso de que el modelo inicial reconozca la categoría 4, mediante umbrales de similitud, se enviará una respuesta para volver a escribir.



Ejemplo

En caso de que se reconozca una categoría entre 1 y 3, el flujo de datos está determinado para optar por una salida pertinente a las clasificaciones previas.



Ejemplo

CONCLUSIONES:

El presente trabajo me implicó hallar una gran cantidad de soluciones en poco tiempo, cubriendo gran parte de los contenidos dictados en la materia, de una manera práctica. El hecho de generar en cierta medida lo que intentaba lograr me brindó satisfacción, más aún al brindar pruebas a colegas que lo han utilizado con sorpresa por su funcionamiento. Al haber cierta distancia y reticencia desde los profesionales de la salud mental (y de otras áreas) respecto al uso de tecnologías IA, me pareció interesante encarar y lograr cierta intersección.

Ya que el Notebook puede ser corrido prácticamente "Stand Alone", existe la posibilidad de trabajar distintos textos de interés en psicoanálisis, para implementar en clases o talleres.

En términos generales el trabajo me fue sumamente demandante, en tanto cantidad y dificultad. En particular inicié el desarrollo con las funciones de LlamaIndex vistas en clase, pero noté que avanzando en ello eran requeridas cuentas pagas de OpenAI, por lo que opté por una implementación más "casera". También podría nombrar que fueron grandes desafíos las implementaciones de las respuestas de la base de datos de grafos y el clasificador de 2 pasos, el cual costó demasiado que abarque y devuelva algo similar a lo pretendido.

Por ello podría decir que se hallan numerosas mejoras en vista, en todas las áreas nombradas.

Librerías utilizadas:

1. **chromadb**
    - ChromaDB en PyPI
2. **PyMuPDF**
    - PyMuPDF en PyPI
3. **langchain**
    - Langchain en PyPI
4. **sentence_transformers**
    - Sentence Transformers en PyPI
5. **transformers**
    - Transformers en PyPI
6. **python-decouple**
    - python-decouple en PyPI
7. **pandas**
    - Pandas en PyPI
8. **tensorflow**
    - TensorFlow en PyPI
9. **tensorflow_hub**
    - TensorFlow Hub en PyPI
10. **tensorflow_text**
    - TensorFlow Text en PyPI
11. **gdown**
    - gdown en PyPI
12. **fuzzywuzzy**
    - fuzzywuzzy en PyPI
13. **SPARQLWrapper**
    - SPARQLWrapper en PyPI

Modelos:

- Zephyr-7b-beta
- MoritzLaurer/mDeBERTa-v3-mnli-xnli

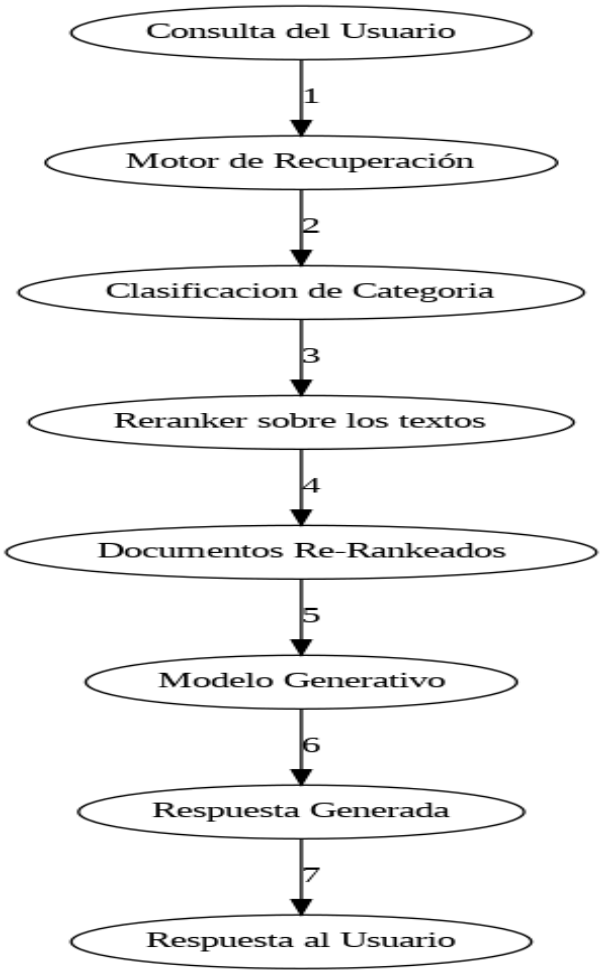
RERANK en modelos RAG:

Por lo que investigué, existen una gran cantidad de desarrollos actuales sobre Rerank, por ejemplo dependientes de LangChain y de VoyageAI. Por lo visto la idea general es lograr la especificidad en las respuestas de los modelos de lenguaje natural. Ya que estos modelos disponen de una gran cantidad de documentos, el Rerank se implementa para dar prioridad a los documentos más importantes mediante una similitud media entre la query y los documentos.

Por ejemplo, LangChain dispone de Cohere reranker, enfocado a interacciones humano-máquina.

Evidentemente busca la coherencia de la respuesta de los modelos de lenguaje, acotando y priorizando lo más adecuado.

Entiendo que en mi trabajo en cuestión, utilizo (no se si lo suficiente) una maniobra similar en lo conceptual con el clasificador Zero-shot sobre las etiquetas de los autores y de los libros, ya que luego de obtener la categoría pertinente, la implementación busca el autor o el libro más pertinente sobre las pocas opciones que el modelo dispone.



Por lo leído, el Rerank se realizaría sobre la totalidad de los documentos, por lo que imagino que debería implementarse luego de, por ejemplo, determinar que la categoría es "1", en ese caso existen chunks de 3 voluminosos libros para comparar con la query, por lo que Rerank sería conveniente para traer lo más pertinente.

Links:

Cohere reranker

MYSCALE