SAT - Temas Avanzados en Deep Learning

Informe Ejercicio 2

Franco Panighini - 61258

Juan Ignacio Matilla - 60459

Santiago Rivas - 61007

Principios de Responsible AI y Safety para Connection RAG

Introducción.

El sistema de sugerencias de RAG aspira a sugerir conexiones profesionales a partir de dossieres generados con LLM y búsqueda vectorial MMR. Dado que incide en oportunidades laborales, convendría alinear su diseño con principios de Responsible AI.

Explainability

Es clave porque los usuarios solo confiarán en un match si entienden su lógica. El sistema podría acompañar cada recomendación con una breve justificación en lenguaje natural que cite las frases del dossier que aportaron mayor similitud, junto con el prompt usado, todo almacenado para auditoría; métricas automáticas de *faithfulness* y revisiones de un *red team* interno ayudarían a detectar alucinaciones; si la explicación incluyera datos sensibles (salud, afinidad política), se reemplazaría por una versión genérica y se mostraría un aviso de privacidad.

Fairness

Resulta vital porque el motor distribuye oportunidades y podría reforzar desigualdades pre-existentes. Antes de indexar, se verificaría el balance demográfico y se filtrarían expresiones discriminatorias; ya en producción, métricas de paridad (p. ej. *disparate impact*) permitirían vigilar la distribución de matches por género, industria o seniority; de detectarse sesgo, se re-ponderarían las similitudes o se ajustarían umbrales, y pruebas A/B ciegas con cohortes diversas evaluarían la percepción de equidad.

Robustness

Pruebas adversarias con texto malicioso en los dossieres comprobarán la resistencia del pipeline de limpieza y del LLM.

Transparency

La transparencia permitirá verificar la lógica de las sugerencias y corregir sesgos a tiempo. Una *model card* pública describiría los datos empleados para los embeddings, la arquitectura general, las limitaciones conocidas y los responsables de mantenimiento; cada recomendación quedaría registrada en un log cifrado que guarde fecha, versión de modelo y parámetros clave; durante el signup del usuario se presentaría una política clara sobre cómo se forman los dossieres y cómo ejercer el derecho de supresión.

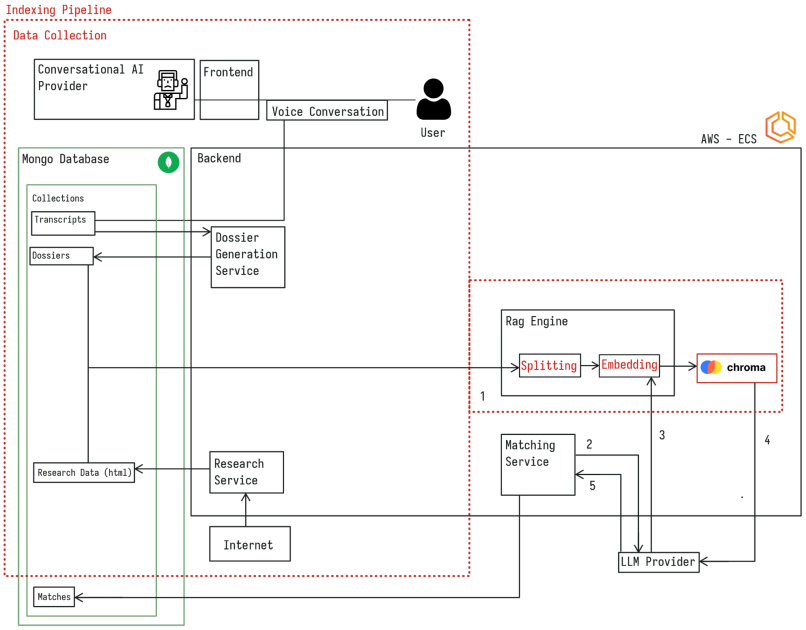
Data Privacy

Es crítica porque se manejan datos personales y reputacionales. Se almacenaría solo la información indispensable para el matchmaking; el usuario elegiría qué información de fuentes públicas incorporar; se podrían cifrar los datos sensibles.

Conclusión

Incorporar principios de buenas prácticas de Responsible AI puede marcar una gran diferencia en cómo los usuarios perciben y adoptan una herramienta. Explicar bien por qué se sugiere una conexión, cuidar qué datos usamos y asegurarnos de que el sistema no refuerce sesgos existentes no solo es importante desde un punto de vista ético, sino que también puede mejorar mucho la experiencia de uso y la confianza general en la plataforma. Logrando que las recomendaciones no sean solo precisas, sino también justas, claras y respetuosas con la información personal de cada usuario.

Modelo en Producción



En la figura se puede observar como el sistema de RAG implementado en este trabajo se podría aplicar en un ambiente de producción. La idea surgió a partir de una arquitectura ya existente y por lo tanto se explicará el sistema y cómo se relaciona con dicha arquitectura.

Los cuadrados rojos muestran las diferentes partes del indexing pipeline. El cuadrado rojo de la derecha es la parte de la arquitectura que se puede considerar como data collection. Luego de que el usuario haya terminado la conversación con la conversational AI, la transcripción de la conversación es guardada en la base de datos Mongo. A partir de estos transcripts se generan los dossier que serán utilizados para el sistema de RAG. Por otro lado, la aplicación principal de backend tendrá un servicio de búsqueda, que a partir de participantes de usuarios realiza un scraping de internet buscando información de dicho participante. Esta información, por ser un scrap estará en html.

A partir de estos datos se realiza el chunking y embedding. Estos pueden utilizar una instancia local como utilizar las herramientas de proveedores como Open AI. En el caso de la arquitectura en producción se utiliza el sistema de embedding se OpenAI. Por razones de testing, en la prueba se utilizó un modelo local para evitar la latencia de los requisitos. Luego los embeddings son cargados en una base de datos vectorial, en este caso se utilizó ChromaDB. Hasta este punto se vio paso 1 del sistema de RAG que es la colección, procesado y carga de los documentos en la base de datos vectorial.

El punto 2 es el llamado al servicio de LLM para realizar los matches dado un usuario peticionado. Para esto se realiza un embedding con el dossier del usuario y se realiza una búsqueda de los chunks más similares a dicho embedding (punto 3 y 4).

Con estos chunks, el LLM podrá generar un matches a partir de los chunks. El LLM además de los chunks, recibirá el nombre e identificador del usuario al que pertenece el chunk. De esta manera el LLM puede identificar como realizar los matches con la información provista. El paso 5 es la respuesta del LLM al servicio de matches. Este servicio guarda los matches generados en la base de datos Mongo.

La arquitectura corre en AWS ECS. El backend corre en Python (FastAPI) mientras que el frontend es servido por un Nginx.