RAG VIDEO RECOMMENDER

01

Problema a resolver

Problema a resolver

- Cuando queremos aprender algo nuevo, la primera opción que tenemos es realizar una búsqueda de tutoriales educativos en Internet. Sin embargo, las respuestas son iguales para todos y no tienen en cuenta nuestra experiencia previa.
- No existe una forma de filtrar por nivel educativo (ejemplo, secundaria, facultad, graduado). Además, hay demasiado contenido autogenerado y de baja calidad que no es posible ignorar en las búsquedas.
- Esto conlleva a que muchas veces exista el contenido, pero no se adapte a nuestras necesidades o no tenga el nivel educativo que buscamos.
- Hay videos más complejos que al no tener tantas vistas no son devueltos en la búsqueda que pueden ser más útiles que otro más básico con más vistas.

Problema a resolver

- Todo esto se traduce en una pérdida de tiempo consumiendo parcialmente los videos para determinar si son útiles o no.

- A la hora de educarnos no alcanza simplemente con filtrar un video por su título o descripción, menos si se trata de un curso de 17 horas. No tenemos forma de saber si lo que realmente queremos aprender se encuentra en ese video.

- En particular los temas de computación, data science, software development o matemática son complejos, varían mucho según el nivel y la experiencia y no siempre hay fuentes curadas que se adecúen al usuario.

Solución:

RAG Video Recommender!

- Búsqueda de videos que son relevantes para el estudiante según su edad, conocimiento previo e intereses.
- Los videos provienen de fuentes con reputación como *Stanford*, *Harvard*, *FreeCodeCamp*, entre otros.
- El usuario ingresa información contextual y explica de forma detallada el contenido que quiere aprender. Ej: "*Quiero aprender cómo definir una clase en Python*".
- La búsqueda se realiza utilizando el contenido real del video a través de matching semántico basado en su transcripción y el video se puede visualizar directamente en la página.
- Se explica al usuario por qué el video recomendado es relevante para él.

02

Desafíos

encontrados

Desafíos encontrados

- Armado del dataset: Se eligieron manualmente playlists de Youtube relevantes de temas como computer science, programación, cloud y matemática.
- Gran cantidad de contenido que tardó 12+ horas en descargarse desde la API de Youtube, y demoró 16+ hs en subir +100k chunks a la base de datos vectorial.

- La carga de datos falló 2 veces luego de un cierto progreso dado que algún chunk no respetaba el formato de subida o porque estaba repetido y se debía comenzar nuevamente. Mejoramos la carga de datos e hicimos un sistema más tolerante a fallas.

Desafíos encontrados

- Algunos videos no tenían el transcript disponible y la API venía vacía por lo que debíamos tomar el tópico en concreto desde otra fuente.
- Una de las métricas más comunes en RAG no aplican exactamente a este caso de uso por lo que hubo que ser descartada luego de numerosos intentos por aplicarla
- Crear datasets de prueba representativos a priori parece una tarea sencilla, pero requiere mucho ingenio y creatividad.
- La API de YouTube no es muy confiable para descargar subtítulos (a veces no toma los autogenerados) por lo que tuvimos que usar yt-dlp.

03

Decisiones de implementación

Decisiones de implementación

- **Query-rewriting** para tomar la descripción del usuario, extraer lo que realmente quiere aprender y hacer un rewrite de la query para llamar a la BD vectorial. Eso se hace porque no se espera que el usuario sea un prompt engineer experto, entonces ir contra la base con la query cruda (donde una buena parte es ruido) puede hacer que la búsqueda sea de peor calidad y que todos los esfuerzos de filtrado y reranking posteriores fueran en vano.

- **Re ranking:** Debido a que el dataset de 4700+ videos, se consultan 15 videos por similitud con los transcripts correspondientes y luego hacer un re-ranking de los más relevantes según el contenido semántico de la query usando un cross-encoder para pasarle al modelo que genera la recomendación final al usuario no más de 5 videos, de forma que no tenga que procesar tanta información y cuide la economía de los tokens.

Decisiones de implementación

- Tomar **3-5 videos más relevantes** y usar la información personal provista por el usuario para mejorar la sugerencia explicándole por qué dichos videos se ajustan a necesidades particulares y mejorar la efectividad del producto.

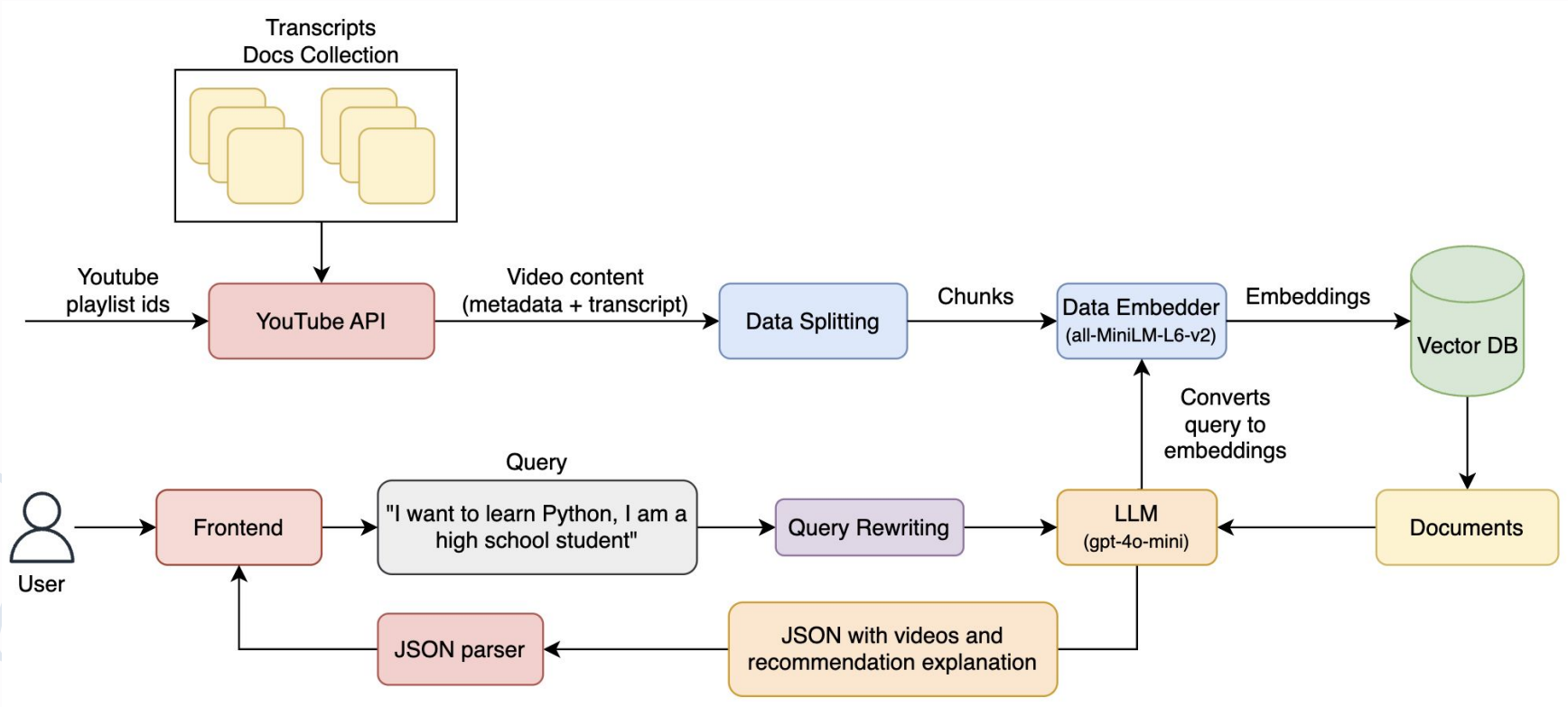
- Estructuramos la respuesta del modelo usando ***Pydantic***: debido a la naturaleza del producto en el que el usuario espera una lista de videos recomendados y no un texto final, se formatea la respuesta del LLM para que pueda ser mostrada en formato cards en el frontend.

Decisiones de implementación

- **Chunking y tokenización**: el guardado en la BD vectorial se realiza tokenizando sus transcripciones con *Tiktoken* y guardando chunks de 200 tokens para poder englobar conceptos enteros de clases dentro de los chunks (hay videos de 17hs)

- Se utiliza un **chunk overlapping** del 20% para que los chunks mantengan la coherencia semántica del documento.

Arquitectura implementada



04

Resultados obtenidos

DEMO

Métricas definidas

Answer Accuracy

Capacidad del producto de encontrar videos relevantes con la query del usuario.

Mira la **salida del LLM:** nombre del video, playlist, justificación de por qué es relevante para el usuario, dificultad.

Context Precisión

Capacidad del retriever para obtener los chunks relevantes por encima de los irrelevantes.

Mira las transcripciones de los chunks obtenidos de la **base vectorizada.**

Context Recall

Cuánto coincide el contexto obtenido con la información verdadera esperada.

Mira las transcripciones de los chunks obtenidos de la **base vectorizada.**

Métrica no aplicada: Faithfulness

Pregunta: ¿Dónde y cuándo nació Einstein?
Contexto: Albert Einstein (nacido el 14 de marzo de 1879) fue un físico teórico alemán, considerado uno de los científicos más importantes e influyentes de todos los tiempos.

Respuesta con alta faithfulness: Einstein nació en Alemania el 14 de marzo de 1879.
Respuesta con baja faithfulness: Einstein nació en Alemania el 20 de marzo de 1879.

Cálculo de la Faithfulness

1. Dividir la respuesta generada en enunciados:
   Enunciado 1: "Einstein nació en Alemania."
   Enunciado 2: "Einstein nació el 20 de marzo de 1879."
2. Verificar si cada enunciado puede inferirse del contexto dado.
   Enunciado 1: Se infiere del contexto
   Enunciado 2: No se infiere del contexto
3. Cálculo de la faithfulness: Fidelidad = 1 / 2 = 0.5

$$\text{Fidelidad} = \frac{\text{de enunciados verdaderos}}{\text{total de enunciados}}$$

Métrica no aplicada: Faithfulness

Interpretación

- Faithfulness = 1.0 → todo el contenido es fiel al contexto.
- Faithfulness = 0.5 → la mitad de los enunciados son correctos.
- Faithfulness = 0.0 → nada del contenido se puede inferir del contexto.

La métrica analiza si cada statement de **salida del LLM** se puede inferir por el contexto obtenido en los chunks → No aplica en este proyecto ya que no se elabora una respuesta fáctica, sino una **lista de sugerencias**. → No se puede dividir en enunciados: el faithfulness es **muy bajo.**

Resultados de la evaluación

Rewriting: False
Reranking: False

	Answer Relevancy	Context Precision	Context Recall
0	0.855027	0.638889	1.000000
1	0.863637	1.000000	1.000000
2	0.891937	0.805556	1.000000
3	0.904335	0.833333	0.666667
4	0.912310	0.583333	1.000000
5	0.896975	0.477778	0.750000

Answer Relevancy (avg): 0.887370
Context Precision (avg): 0.723148
Context Recall (avg): 0.92778

Resultados de la evaluación

Rewriting: False
Reranking: True

	Answer Relevancy	Context Precision	Context Recall
0	0.863988	0.0	1.000000
1	0.888786	1.000000	1.000000
2	0.886745	1.000000	0.750000
3	0.902932	1.000000	0.666667
4	0.907005	0.5	0.333333
5	0.897638	0.5	0.250000

Answer Relevancy (avg): 0.891182
Context Precision (avg): 0.666667
Context Recall (avg): 0.666667

Resultados de la evaluación

Rewriting: True
Reranking: True

	Answer Relevancy	Context Precision	Context Recall
0	0.856859	0.0	0.500000
1	0.886791	1.000000	1.000000
2	0.886916	1.000000	0.750000
3	0.912313	1.000000	0.666667
4	0.921207	0.5	0.333333
5	0.907857	0.5	0.750000

Answer Relevancy (avg): 0.895324
Context Precision (avg): 0.666667
Context Recall (avg): 0.666667

Resultados de la evaluación

Rewriting: True
Reranking: False

Mejor Combinación

	Answer Relevancy	Context Precision	Context Recall
0	0.861534	0.477778	1.000000
1	0.855887	1.000000	1.000000
2	0.889354	0.804167	1.000000
3	0.908314	0.805556	1.000000
4	0.888200	0.679167	1.000000
5	0.907857	0.679167	0.750000

Answer Relevancy (avg): 0.884239
Context Precision (avg): 0.740972
Context Recall (avg): 0.958333

05

Conclusiones

Conclusiones

- Para este sistema en particular, resulta efectivo aplicar query rewriting pero hacer query reranking no brinda ningún beneficio según muestran los resultados de las pruebas
- El RAG es un sistema que escala bien para grandes volúmenes de datos, extender nuestro trabajo a otras áreas seria solo cuestion de obtener un dataset mas grande.
- El pipeline presentado depende fuertemente de la calidad de los datos provistos, lo cual conlleva un trabajo de curaduría no trivial.
- Se necesita incorporar en la fuente de datos el mismo contenido con distintos grados de dificultad para que este sistema responda adecuadamente a los distintos perfiles de usuarios.

¡GRACIAS!

¿PREGUNTAS?