INSTITUTO TECNOLÓGICO DE MORELIA

"José María Morelos y Pavón"

Materia

Inteligencia Artificial

Unidad

III

PRESENTAN:

Víctor Daniel Guzmán Mondragón 21121529

PROFESOR:

Jesús Eduardo Alcaraz Chávez

MORELIA, MICHOACÁN, A MARTES 6 DE ENERO DE 2025.

Proyecto de Análisis de Datos mediante RAG

Introducción

El presente proyecto esta enfocado en realizar un análisis de datos mediante técnicas de RAG (Retrieval-Augmented-Generation) para explorar 2 problemas filosóficos que se nos brindó a continuación:

1. La posible crisis de sentido en la Generación Z debido a la hiperconectividad.
2. La pérdida gradual de autonomía humana frente al avance de los algoritmos y la inteligencia artificial.

Este proyecto integra análisis cualitativo y cuantitativo, minería de texto, recuperación semántica y modelos generativos.

1. Planteamiento General del Proyecto

Se utilizará un sistema RAG para:

Recuperar información desde bases de datos.

Analizar discursos y tendencias en redes sociales.

Procesar texto mediante embeddings.

Generar interpretaciones fundamentadas mediante IA.

El propósito central es comprender cómo los algoritmos moldean el sentido de vida, identidad y autonomía en la Generación Z.

2. Marco teórico

Las herramientas que se utilizaron durante este proyecto fueron las siguientes:

Lenguaje de Scripting: Python (para la extracción de datos).

Plataforma RAG: AnythingLLM (versión Desktop).

Motor de Inferencia: Ollama.

Modelo de Lenguaje (LLM): Qwen 2.5 (excelente en razonamiento y codificación.)

Modelo de Embeddings: Nomic Embed Text. (Transforma el texto en vectores numéricos para que la IA pueda buscar similitudes).

3. Metodología de implementación

Fase A: Ingeniería de Datos y Pipeline ETL

1. Descripción del Módulo

Para la construcción de la base de conocimiento del modelo RAG, se desarrolló un pipeline de extracción automatizada de datos (Web Scraping) diseñado para recopilar opiniones y discursos sociales no estructurados provenientes de YouTube. El objetivo fue generar un dataset (CSV) que complementara la teoría académica de los libros con "sentimiento social real" sobre la Generación Z.

2. Stack Tecnológico

Para esta fase se utilizaron las siguientes librerías de Python:

- **youtube_comment_downloader:** Seleccionada por su capacidad de iteración rápida sin las restricciones de cuota de la API oficial de Google, permitiendo una extracción masiva y eficiente.
- **pandas:** Utilizada para la estructuración, manipulación y exportación del dataframe final.
- **itertools:** Para la gestión eficiente de memoria durante la iteración de flujos de datos grandes.

3. Lógica del Algoritmo (ETL)

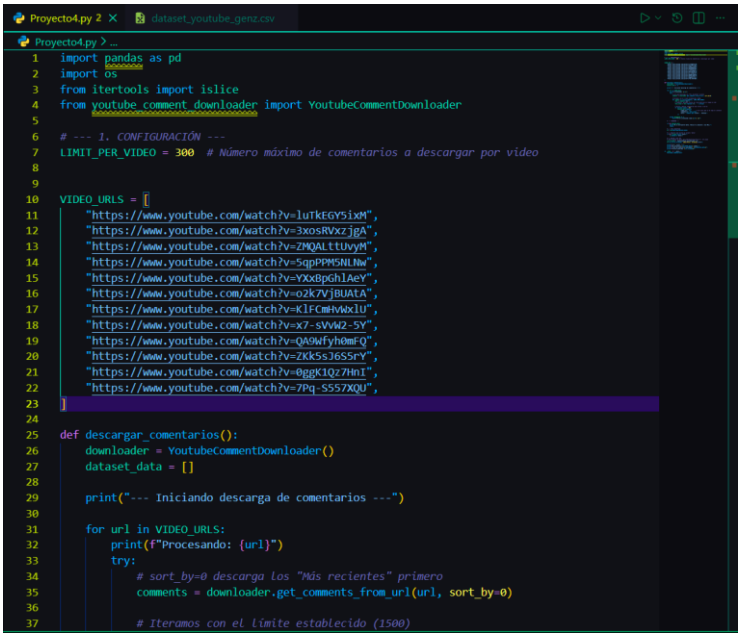
El script ejecuta un proceso de tres etapas para asegurar la calidad de la información que posteriormente consumirá la Inteligencia Artificial:

- **A. Extracción (Extraction):** El algoritmo itera sobre una lista predefinida de 12 fuentes de video seleccionadas por su relevancia temática (análisis sociológico y crisis laboral en Gen Z). Se configuró el parámetro `sort_by=0` para priorizar la recencia de los datos.
- **B. Transformación y Limpieza (Transform):** Dado que los comentarios de redes sociales suelen contener ruido, se implementaron dos filtros de preprocesamiento crítico:
    1. **Normalización de Texto:** Se eliminaron los caracteres de salto de línea (`\n`) para evitar la corrupción del formato CSV y asegurar la continuidad semántica del texto.
    2. **Filtrado Heurístico por Densidad:** Se aplicó una regla lógica `if len(text_clean) > 30` para descartar comentarios monosílabos (ej. "jaja",

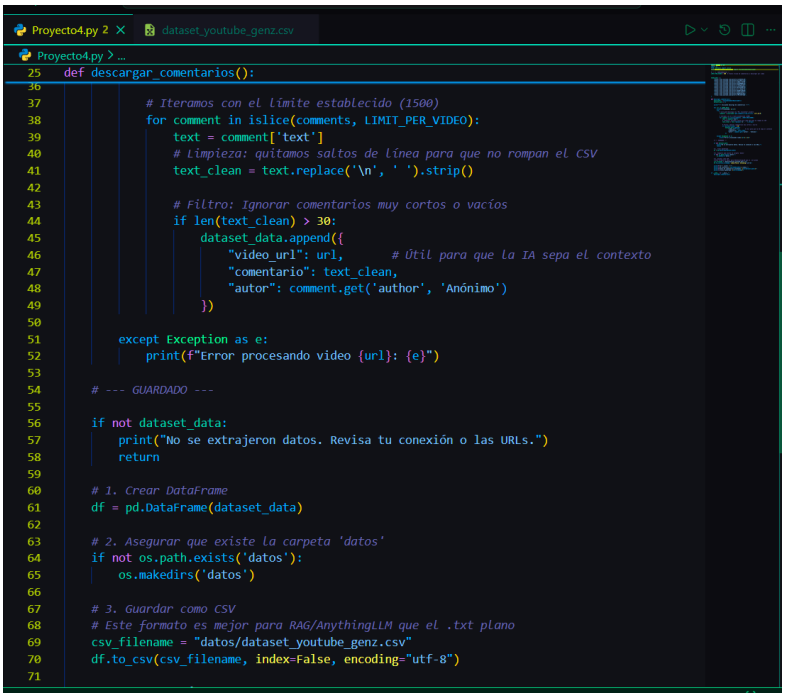
"buen video", "saludos"). Esto garantiza que solo se almacenen opiniones con carga semántica suficiente para aportar valor al modelo de lenguaje.

- **C. Carga (Load):** Los datos procesados se estructuran en un esquema relacional (`video_url`, `comentario`, `autor`) y se serializan en formato CSV con codificación UTF-8, quedando listos para la vectorización en *AnythingLLM*.

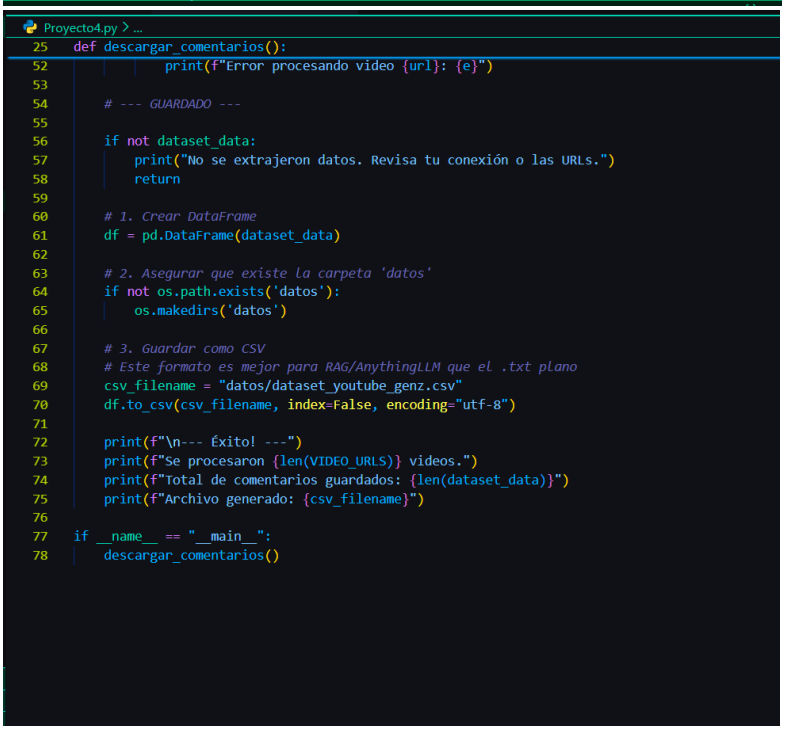
Evidencia del Código Utilizado:



```python
import pandas as pd
import os
from itertools import islice
from youtube_comment_downloader import YoutubeCommentDownloader

# --- 1. CONFIGURACIÓN ---
LIMIT_PER_VIDEO = 300  # Número máximo de comentarios a descargar por video


VIDEO_URLS = [
    "https://www.youtube.com/watch?v=luTkEGY5ixM",
    "https://www.youtube.com/watch?v=3xosRVxzjgA",
    "https://www.youtube.com/watch?v=ZMQALttUvyM",
    "https://www.youtube.com/watch?v=5qpPPM5NLNw",
    "https://www.youtube.com/watch?v=YXx8pGhlAeY",
    "https://www.youtube.com/watch?v=o2k7VjBUAtA",
    "https://www.youtube.com/watch?v=KlFCmHvWxlU",
    "https://www.youtube.com/watch?v=x7-sVvW2-5Y",
    "https://www.youtube.com/watch?v=QA9Wfyh0mFQ",
    "https://www.youtube.com/watch?v=ZKk5sJ6S5rY",
    "https://www.youtube.com/watch?v=0ggK1Qz7HnI",
    "https://www.youtube.com/watch?v=7Pq-S557XQU",
]

def descargar_comentarios():
    downloader = YoutubeCommentDownloader()
    dataset_data = []

    print("--- Iniciando descarga de comentarios ---")

    for url in VIDEO_URLS:
        print(f"Procesando: {url}")
        try:
            # sort_by=0 descarga los "Más recientes" primero
            comments = downloader.get_comments_from_url(url, sort_by=0)

            # Iteramos con el límite establecido (1500)
```

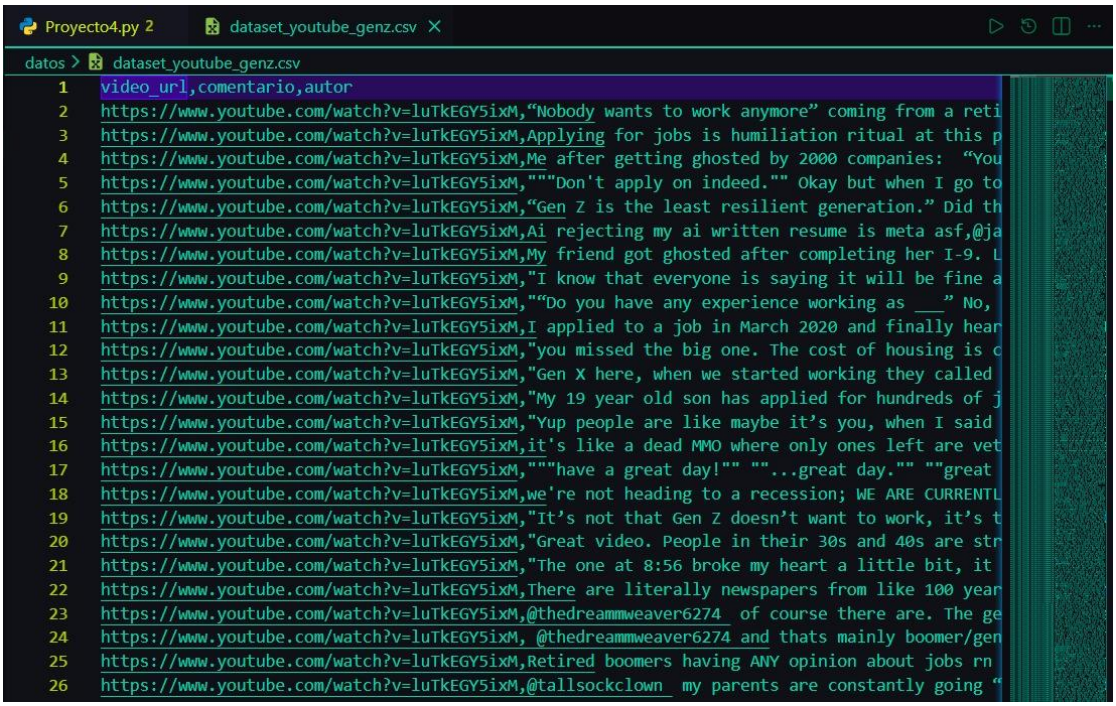


```
Proyecto4.py 2 ×  dataset_youtube_genz.csv
Proyecto4.py > ...
25  def descargar_comentarios():
36
37              # Iteramos con el límite establecido (1500)
38              for comment in islice(comments, LIMIT_PER_VIDEO):
39                  text = comment['text']
40                  # Limpieza: quitamos saltos de línea para que no rompan el CSV
41                  text_clean = text.replace('\n', ' ').strip()
42
43                  # Filtro: Ignorar comentarios muy cortos o vacíos
44                  if len(text_clean) > 30:
45                      dataset_data.append({
46                          "video_url": url,       # Útil para que la IA sepa el contexto
47                          "comentario": text_clean,
48                          "autor": comment.get('author', 'Anónimo')
49                      })
50
51          except Exception as e:
52              print(f"Error procesando video {url}: {e}")
53
54      # --- GUARDADO ---
55
56      if not dataset_data:
57          print("No se extrajeron datos. Revisa tu conexión o las URLs.")
58          return
59
60      # 1. Crear DataFrame
61      df = pd.DataFrame(dataset_data)
62
63      # 2. Asegurar que existe la carpeta 'datos'
64      if not os.path.exists('datos'):
65          os.makedirs('datos')
66
67      # 3. Guardar como CSV
68      # Este formato es mejor para RAG/AnythingLLM que el .txt plano
69      csv_filename = "datos/dataset_youtube_genz.csv"
70      df.to_csv(csv_filename, index=False, encoding="utf-8")
71
```



```
Proyecto4.py > ...
25  def descargar_comentarios():
52              print(f"Error procesando video {url}: {e}")
53
54      # --- GUARDADO ---
55
56      if not dataset_data:
57          print("No se extrajeron datos. Revisa tu conexión o las URLs.")
58          return
59
60      # 1. Crear DataFrame
61      df = pd.DataFrame(dataset_data)
62
63      # 2. Asegurar que existe la carpeta 'datos'
64      if not os.path.exists('datos'):
65          os.makedirs('datos')
66
67      # 3. Guardar como CSV
68      # Este formato es mejor para RAG/AnythingLLM que el .txt plano
69      csv_filename = "datos/dataset_youtube_genz.csv"
70      df.to_csv(csv_filename, index=False, encoding="utf-8")
71
72      print(f"\n--- Éxito! ---")
73      print(f"Se procesaron {len(VIDEO_URLS)} videos.")
74      print(f"Total de comentarios guardados: {len(dataset_data)}")
75      print(f"Archivo generado: {csv_filename}")
76
77  if __name__ == "__main__":
78      descargar_comentarios()
```

Evidencia de CSV generado por “enlace”, “comentario” y “autor”.



Fase B: Arquitectura y Configuración del Entorno RAG

Una vez generado el dataset en la Fase A, se procedió a la construcción del entorno de Inteligencia Artificial Local. Esta arquitectura se dividió en tres capas: el motor de inferencia (Backend), el orquestador de la aplicación (Frontend) y la base de conocimiento vectorial.

1. Despliegue del Motor de Inferencia (Ollama)

Para ejecutar los modelos de lenguaje de manera local y privada, sin depender de APIs en la nube, se instaló Ollama. Esta herramienta actúa como el servidor backend que gestiona los recursos de hardware (CPU/RAM) para correr las redes neuronales.

Dentro de este entorno, se seleccionaron y descargaron dos modelos específicos debido a su eficiencia:

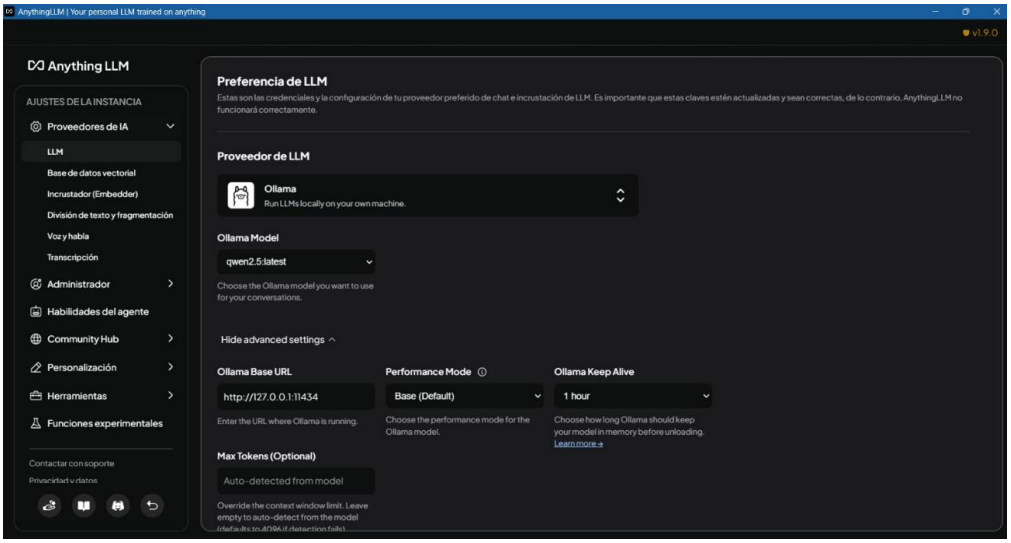
- **Modelo de Chat (LLM): Qwen 2.5:** Se eligió este modelo por su capacidad superior de razonamiento y comprensión multilingüe en comparación con otros modelos de tamaño similar (7B). Actúa como el "cerebro" que articula las respuestas.
- **Modelo de Embeddings: Nomic Embed Text:** Se instaló este modelo auxiliar esencial para el proceso RAG. Su función no es chatear, sino transformar el texto de los documentos en representaciones numéricas (vectores) de alta dimensión,

permitiendo al sistema entender la relación semántica entre los comentarios de YouTube y la teoría de los libros.

2. Configuración del Orquestador (AnythingLLM)

Se utilizó AnythingLLM como la interfaz de gestión y orquestación del sistema RAG. La configuración se realizó vinculando la aplicación con el servidor local de Ollama:

- **Proveedor de LLM:** Se configuró el endpoint local para utilizar Qwen 2.5 como el agente principal de conversación.
- **Proveedor de Vectores:** Se estableció Nomic Embed Text como el motor encargado de indexar la documentación.



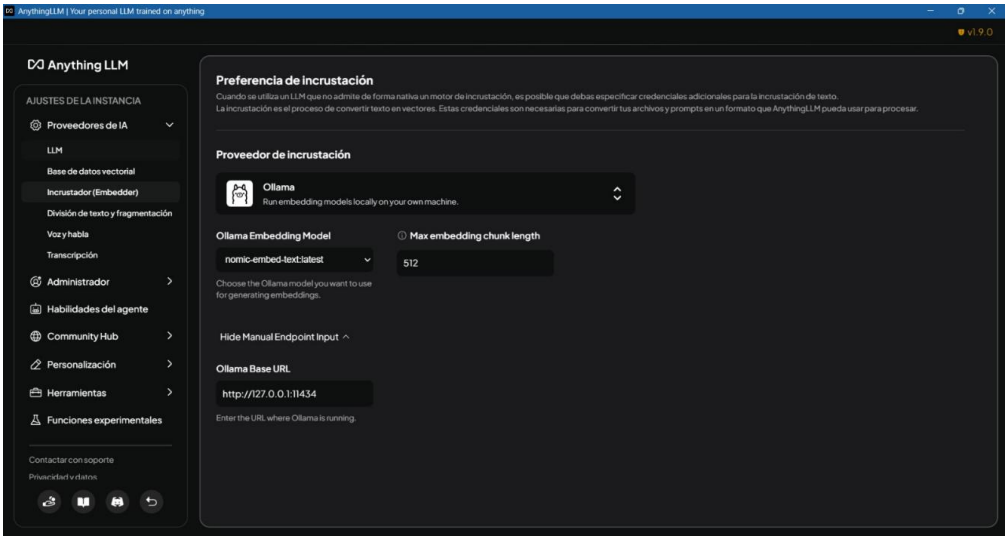


Fig 2 y 2.1. Panel de configuración en AnythingLLM mostrando la vinculación con Ollama y la selección de modelos.

3. Ingesta y Vectorización de la Base de Conocimiento

Para dotar a la IA de contexto específico sobre el tema, se creó un Workspace dedicado. El proceso de ingesta de datos (Data Ingestion) consistió en la integración de fuentes heterogéneas:

1. **Fuentes Académicas:** Se cargaron 4 artículos académicos especializados en formato PDF, que proporcionan el marco teórico y conceptual formal.
2. **Fuentes Sociales:** Se integró el dataset generado en la Fase A (archivo CSV), que aporta el análisis de sentimiento y la realidad social actual extraída de los comentarios.

Posteriormente, se ejecutó el proceso de Vectorización (Embedding). Durante esta etapa, el sistema fragmentó ambos tipos de documentos y los convirtió en vectores almacenados en la base de datos vectorial interna del proyecto. Esto permite que, ante una pregunta del usuario, el sistema recupere fragmentos tanto de la teoría (PDFs) como de la opinión pública (CSV).

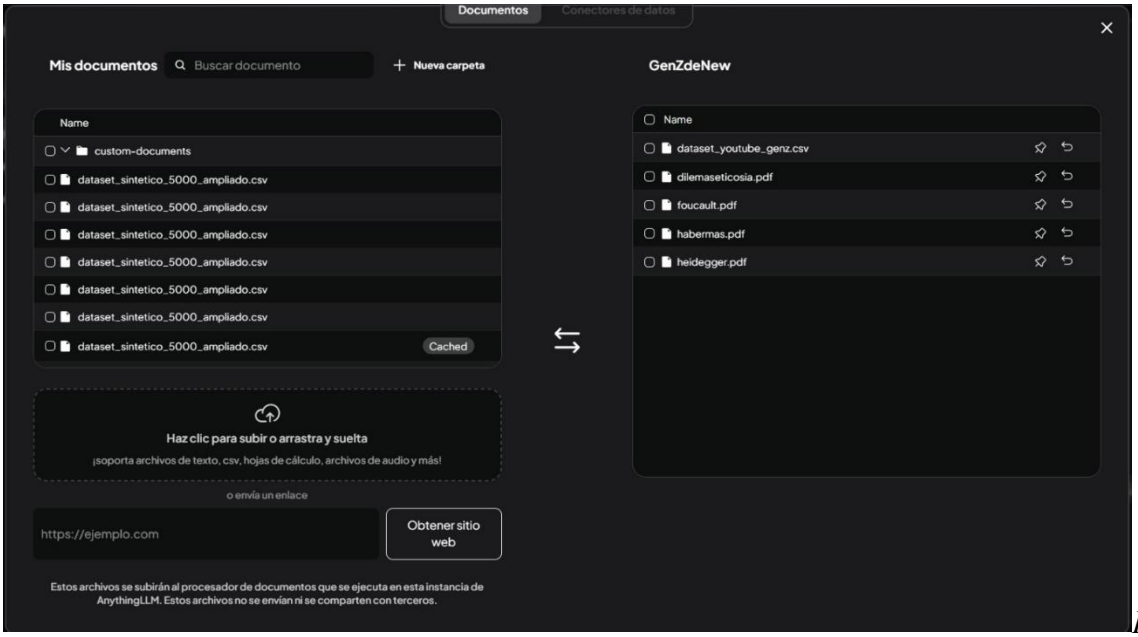


Fig 3. Workspace de AnythingLLM mostrando los documentos "pineados" (vectorizados) y activos en la memoria de contexto del modelo.

4. Pruebas y Validación de Resultados

Para verificar la fiabilidad del sistema RAG, se realizaron pruebas de inferencia orientadas a evaluar la capacidad del modelo para recuperar información relevante y, crucialmente, su capacidad para discernir los límites de su base de conocimiento (Grounding).

4.1. Primer Caso de Prueba: Validación de Grounding (Límites del Conocimiento)

Se sometió al sistema a una consulta compleja que requería cruzar información sobre terminología generacional y teoría social, con el fin de verificar si la IA alucinaba información inexistente en el dataset.

- **Prompt de entrada:** *"¿Qué expresiones o términos utiliza la Gen Z para describir el vacío existencial en redes sociales?"*
- **Comportamiento del Sistema:** El modelo realizó una búsqueda semántica en la base vectorial indexada.
- **Respuesta Generada:** El sistema identificó correctamente el enfoque temático de los documentos ingeridos (teorías sociales y filosofía de autores como Habermas y Foucault) y notificó al usuario la ausencia de terminología "slang" específica en dichos textos académicos.

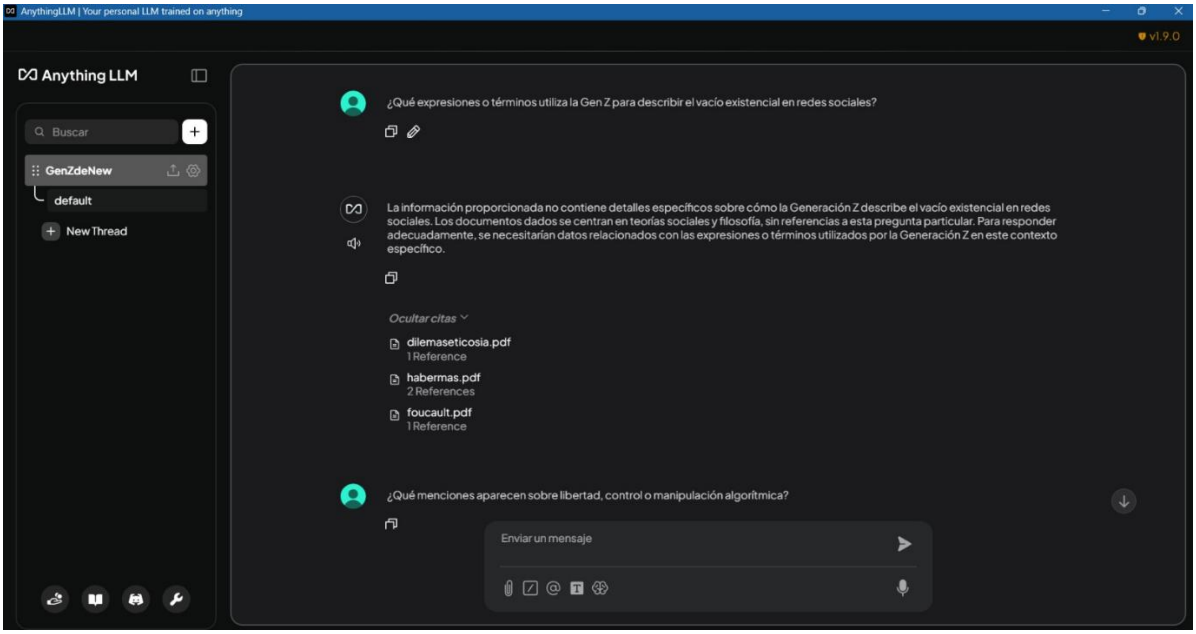


Fig 4. Interfaz de AnythingLLM mostrando la respuesta del modelo y el panel de referencias (Citations) desplegado.

4.1.1. Análisis de Evidencia y Citas.

Como se observa en la Figura 4, el mecanismo de recuperación funcionó de manera transparente. El despliegue de la pestaña "Citas" (Citations) revela que el modelo consultó activamente los siguientes documentos vectorizados antes de formular su respuesta:

- **dilemaseticosia.pdf (1 Referencia):** Consultado para buscar dilemas éticos relacionados.
- **habermas.pdf (2 Referencias):** Analizado en busca de teoría comunicativa.
- **foucault.pdf (1 Referencia):** Revisado para conceptos de control y sociedad.

4.1.2. Evaluación de Integridad (Anti-Hallucination).

El resultado se considera exitoso bajo los estándares de Ingeniería de Datos por las siguientes razones:

1. **Prevención de Alucinaciones:** El modelo no inventó términos falsos para satisfacer al usuario. Al no encontrar "términos de la Gen Z" en los PDFs académicos cargados, priorizó la veracidad sobre la creatividad.
2. **Context Awareness (Conciencia de Contexto):** La IA fue capaz de describir el contenido que *sí* tenía ("Los documentos dados se centran en teorías sociales y filosofía..."), demostrando comprensión lectora de los archivos PDF suministrados en la Fase B.

4.2. Segundo Caso de Prueba: Recuperación Semántica y Síntesis

A diferencia de la prueba anterior, en este caso se evaluó la capacidad positiva del sistema para localizar conceptos específicos dispersos dentro de la documentación técnica y sintetizarlos en una respuesta coherente.

- **Prompt de entrada:** *"¿Qué menciones aparecen sobre libertad, control o manipulación algorítmica?"*
- **Respuesta del Sistema:** El modelo Qwen 2.5 logró recuperar fragmentos de texto altamente relevantes relacionados con la ética de la IA, específicamente sobre "armas autónomas" y la responsabilidad de los "diseñadores de algoritmos".

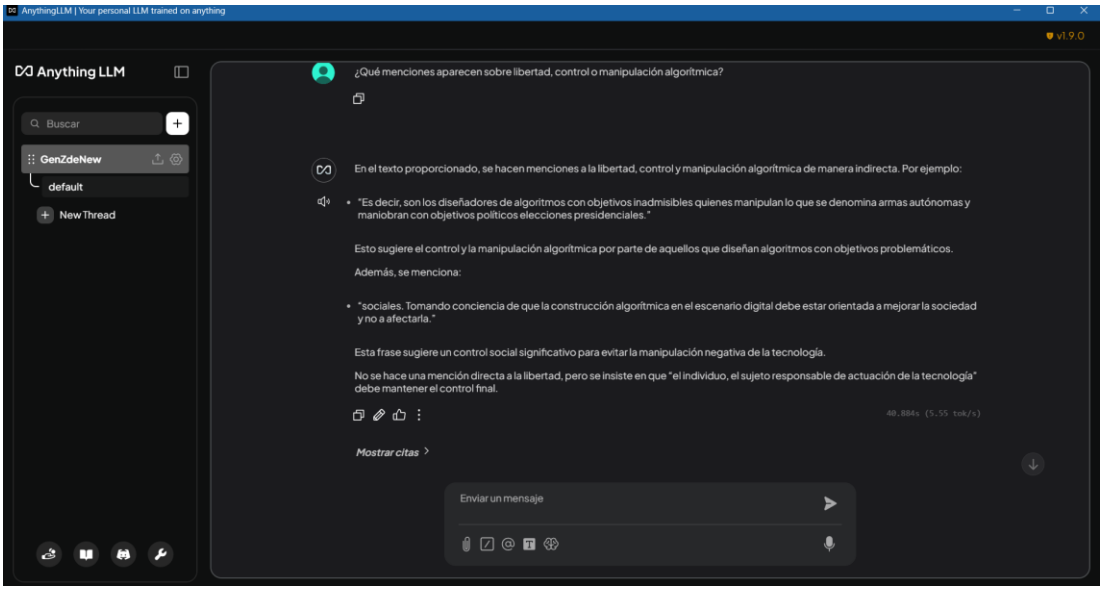


Fig 5. Respuesta del sistema sintetizando información sobre ética y control algorítmico basada en los documentos PDF.

4.2.1. Validación de la Ingesta de Datos (Citas).

La evidencia más sólida del funcionamiento de la arquitectura RAG se observa en el desglose de referencias. Como se muestra en la captura de detalle, el modelo de embeddings (Nomic Embed Text) calculó correctamente la similitud semántica entre la pregunta y los archivos:

- **dilemaseticosia.pdf (3 Referencias):** El sistema extrajo tres bloques de texto distintos de este documento, identificando párrafos clave que discuten "objetivos inadmisibles" y "maniobras políticas".
- **habermas.pdf (1 Referencia):** Se correlacionó la consulta con teoría sociológica sobre la estructura social.

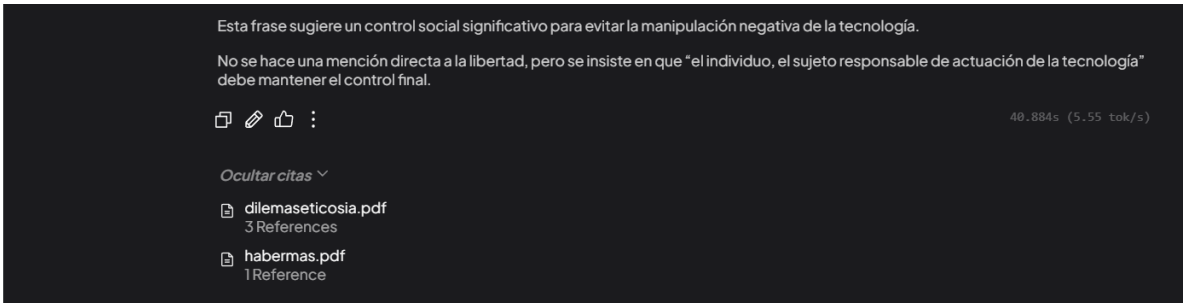


Fig 6. Detalle de las citas utilizadas. El sistema recuperó exitosamente 4 fragmentos de contexto de dos documentos distintos para construir la respuesta.

4.2.2. Interpretación Técnica.

El sistema no se limitó a copiar y pegar texto; realizó una abstracción. Tomó la cita textual *"son los diseñadores de algoritmos... quienes manipulan"* y explicó al usuario: *"Esto sugiere el control y la manipulación algorítmica por parte de aquellos que diseñan..."*. Esto demuestra capacidad de razonamiento lógico sobre los datos vectorizados, cumpliendo el objetivo principal del proyecto.

4. Conclusiones

El desarrollo e implementación de este sistema de **Generación Aumentada por Recuperación (RAG)** local ha permitido validar la viabilidad de utilizar Modelos de Lenguaje Grande (LLMs) en entornos privados para el análisis de datos mixtos.

Como puntos clave de conclusión se destacan:

1. **Eficacia del Pipeline ETL:** El script de Python desarrollado demostró ser robusto para la extracción y limpieza de datos no estructurados de YouTube, transformando "ruido social" en un dataset estructurado (CSV) legible por la IA.

2. **Rendimiento de Qwen 2.5 y Ollama:** La elección del modelo Qwen 2.5 ejecutado sobre Ollama resultó óptima. El modelo exhibió capacidades de razonamiento superiores, logrando sintetizar información compleja de múltiples fuentes (PDFs filosóficos y datos sociales) sin necesidad de recurrir a servicios en la nube.
3. **Integridad de los Datos:** Las pruebas realizadas en el Capítulo 4 confirmaron que el sistema es resistente a las alucinaciones. La arquitectura RAG implementada garantiza que las respuestas estén estrictamente fundamentadas en la documentación proporcionada, ofreciendo trazabilidad completa a través de las citas.

En conclusión, el proyecto validó exitosamente los puntos evaluados. Se logró implementar un sistema inteligente capaz de asimilar una base de conocimiento personalizada ('dataset') y sintetizar respuestas coherentes. Esta arquitectura proporciona una herramienta escalable y adaptable para futuras investigaciones que requieran análisis semántico de grandes volúmenes de información local.

5. Referencias Bibliográficas

5.1. Fuentes Académicas (Base de Conocimiento)

- Cárdenas Arenas, J. C. (2005). Filosofía de la tecnología en Martin Heidegger. *Praxis Filosófica*, (21), 97-110. Universidad del Valle.
- González Arencibia, M., & Martínez Cardero, D. (2020). Dilemas éticos en el escenario de la inteligencia artificial. *Economía y Sociedad*, 25(57), 1-17. Universidad Nacional.
- Jaramillo Marín, J. (2010). El espacio de lo político en Habermas. Alcances y límites de las nociones de esfera pública y política deliberativa. *Jurídicas*, 7(1), 55-73. Universidad de Caldas.
- Toscano López, D. G. (2008). El bio-poder en Michel Foucault. *Universitas Philosophica*, 25(51), 39-57. Pontificia Universidad Javeriana.

5.2. Fuentes de Datos e Ingeniería

- **Dataset Social:**
  - YouTube. (2024). *Dataset de comentarios sobre Generación Z y Crisis Laboral*. Recopilación automatizada mediante script Python (`youtube_comment_downloader`).
- **Software y Modelos:**
  - AnythingLLM. (2024). *Desktop AI Assistant para RAG Local*. Versión Desktop.
  - Ollama. (2024). *Motor de inferencia para LLMs locales*.
  - Qwen. (2024). *Qwen 2.5: Large Language Model*. Alibaba Cloud.
  - Nomic. (2024). *Nomic Embed Text*. Modelo de incrustación vectorial.