INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

DIPLOMADO EN INTELIGENCIA ARTIFICIAL

Asistente de Investigación con IA Generativa

P R E S E N T A:

CASAS MEJÍA HUGO ALFONSO

Profesor:

M. en C. ALAN BADILLO SALAS

Ciudad de México septiembre 2025

Contenido

Resumen

En el ámbito de la investigación académica, el proceso de revisión de literatura es fundamental, pero a menudo lento y abrumador debido al creciente volumen de publicaciones. Este proyecto presenta el desarrollo de un Asistente de Investigación Híbrido, una herramienta de software diseñada para automatizar y acelerar este proceso. La herramienta utiliza un modelo híbrido que combina la búsqueda de artículos relevantes en bases de datos en línea, a través de la API de Semantic Scholar, con el análisis de documentos PDF proporcionados por el usuario. Mediante el uso de un Modelo de Lenguaje Grande (LLM), específicamente Gemini 1.5 Flash, el sistema realiza un análisis crítico consolidado del material. Como caso de estudio, se analizó el tema de "Exoesqueletos", generando exitosamente un reporte estructurado que identifica el estado del arte, detecta brechas de investigación y propone nuevas líneas de investigación. Se concluye que el uso de LLMs a través de un enfoque híbrido es una estrategia viable y eficiente para potenciar las capacidades de investigación, produciendo reportes detallados y coherentes.

1. Introducción

La presente sección introduce el proyecto de investigación y desarrollo de un Asistente de Investigación Híbrido. Se comienza por contextualizar el desafío que representa la revisión de literatura en el entorno académico actual, estableciendo la justificación y el problema a resolver. Posteriormente, se define de manera clara el objetivo general que guio el proyecto y se desglosan los objetivos específicos que marcaron la ruta para su implementación y validación.

1.1. Justificación y Planteamiento del Problema

La investigación científica y el desarrollo tecnológico se fundamentan en un profundo entendimiento del conocimiento existente. Este proceso, conocido como revisión de literatura o estado del arte, implica la búsqueda, lectura y análisis crítico de decenas o cientos de publicaciones académicas. Tradicionalmente, esta es una tarea manual, intensiva en tiempo y esfuerzo, que puede representar una barrera significativa para la innovación. La sobrecarga de información, con miles de artículos publicados diariamente, agrava este desafío, haciendo cada vez más difícil para los investigadores y estudiantes mantenerse actualizados e identificar áreas donde puedan hacer una contribución original. La llegada de los Modelos de Lenguaje Grandes (LLMs) presenta una oportunidad única para reimaginar y automatizar este proceso.

1.2. Objetivo General

Desarrollar y validar una herramienta de software funcional que automatiza el análisis de literatura científica para identificar el estado del arte y las brechas de investigación, utilizando un Modelo de Lenguaje Grande y un enfoque híbrido de recopilación de datos.

1.3. Objetivos Específicos

- Implementar un módulo en Python para la extracción de texto de documentos en formato PDF.

- Integrar la API de Semantic Scholar para realizar búsquedas programáticas de artículos científicos relevantes.

- Diseñar un prompt estructurado ("Prompt Maestro Híbrido") capaz de guiar a un LLM para realizar un análisis crítico y consolidado.

- Desarrollar un script principal que orqueste el flujo de trabajo, desde la entrada del usuario hasta la generación de un reporte final en formato de texto.

2. Metodología y Herramientas

La selección de una metodología adecuada y las herramientas tecnológicas correctas fue un paso crucial para garantizar el éxito del proyecto. En esta sección se describe la arquitectura estratégica del sistema, así como el conjunto de tecnologías y librerías de software que permitieron su implementación.

2.1. Arquitectura del Sistema: Un Modelo Híbrido

La solución propuesta se basa en una arquitectura de modelo híbrido que combina las fortalezas de la búsqueda automatizada por IA y el conocimiento específico del usuario. Este enfoque colaborativo asegura una cobertura más completa y profunda del tema de investigación. El flujo de trabajo es el siguiente: el usuario provee un tema; el sistema busca artículos en línea y procesa los PDFs que el usuario ya posee; finalmente, toda la información se consolida y se envía al LLM para un análisis unificado.

2.2. Tecnologías Utilizadas

- **Lenguaje de Programación:** Python 3.

- **Editor de Código:** Sublime Text 3.

- **API de IA Generativa:** Se utilizó la API de Google Gemini, específicamente el modelo gemini-1.5-flash-latest, por su balance entre velocidad, costo y capacidad de análisis.
- **API de Búsqueda Académica:** Se integró la API de Semantic Scholar para la búsqueda y recuperación de metadatos de artículos científicos.
- **Librerías de Python:**
    - google-generativeai: Para la interacción con la API de Gemini.
    - requests: Para realizar las solicitudes a la API de Semantic Scholar.
    - PyMuPDF: Para una extracción de texto eficiente y confiable de los archivos PDF.

3. Desarrollo e Implementación

La construcción del Asistente de Investigación se llevó a cabo siguiendo un enfoque modular, dividiendo el sistema en componentes funcionales e independientes. A continuación, se detalla la implementación técnica de cada módulo, así como el diseño del prompt que constituye el núcleo de la inteligencia de la herramienta.

3.1. Módulo de Procesamiento de PDFs (pdf_processor.py)

Este módulo contiene la función extraer_texto_de_pdf, cuya única responsabilidad es recibir la ruta de un archivo PDF y devolver su contenido textual completo. Se utilizó la librería PyMuPDF por su alto rendimiento y precisión en la extracción de texto, incluso en documentos con diseños complejos.

3.2. Módulo de Búsqueda de Artículos (article_finder.py)

Este script implementa la función buscar_articulos, que se conecta a la API de Semantic Scholar. Recibe un tema de investigación y devuelve una lista estructurada con los metadatos de los artículos más relevantes, incluyendo título, autores, año y resumen.

3.3. Orquestación y Lógica Principal (main.py)

Este es el script principal que el usuario ejecuta. Orquesta todo el proceso:

1. Solicita al usuario el tema de investigación.
2. Llama a buscar_articulos para obtener la data de la API.
3. Escanea la carpeta local articulos y llama a extraer_texto_de_pdf por cada archivo encontrado.
4. Construye el Prompt Maestro Híbrido.
5. Envía la solicitud final a la API de Gemini.
6. Llama a la función guardar_reporte para almacenar el resultado.

3.4. Diseño del Prompt Híbrido

El corazón del sistema reside en el diseño del prompt. Se diseñó una plantilla detallada que le da al LLM un rol específico (Investigador Académico Senior), un contexto claro, las fuentes de información separadas (resúmenes de la API y textos completos locales) y una tarea estructurada para la salida.

master_prompt_hibrido = f"""

Actúa como un Investigador Académico Senior y experto en {tema}. Mi objetivo es escribir un artículo de revisión sobre este tema y necesito una base sólida.

He realizado una doble recopilación de información. Primero, una búsqueda automática que encontró los siguientes artículos (de los cuales solo tengo el resumen):

[ARTÍCULOS ENCONTRADOS ONLINE (RESÚMENES)]

{info_api}

[FIN DE ARTÍCULOS ONLINE]

Adicionalmente, tengo una colección personal de artículos en PDF, de los cuales te proporciono el texto completo a continuación:

[ARTÍCULOS LOCALES (TEXTO COMPLETO)]

{info_local}

[FIN DE ARTÍCULOS LOCALES]

[TAREA]

Basado en el análisis CONSOLIDADO de TODA la información (tanto los resúmenes online como los textos completos locales), genera un reporte de investigación que siente las bases para un nuevo artículo. El reporte debe estar en formato Markdown con las siguientes secciones:

1. Síntesis del Estado del Arte

2. Identificación de Brechas y Contradicciones (Gap Analysis)

3. Propuesta de Tesis para un Nuevo Artículo

"""

4. Resultados: Caso de Estudio "Exoesqueletos"

Para validar la funcionalidad y la calidad del análisis de la herramienta desarrollada, se realizó un caso de estudio práctico. Se seleccionó un tema de investigación actual y relevante para ejecutar el flujo de trabajo completo, desde la búsqueda de información hasta la generación del reporte final. Esta sección presenta el proceso y los resultados obtenidos.

4.1. Proceso de Ejecución

Para validar la herramienta, se realizó un caso de estudio con el tema "Exoesqueletos". El sistema fue ejecutado y se le proporcionó dicho tema. Automáticamente, buscó 5 artículos en Semantic Scholar y procesó 3 PDFs que se encontraban en la carpeta local articulos. Toda esta información fue consolidada y enviada a Gemini para su análisis.

4.2. Reporte Generado

El sistema generó de manera exitosa el siguiente reporte consolidado, el cual fue guardado en un archivo de texto.

Reporte de Investigación: Exoesqueletos

Este reporte sintetiza la información proporcionada, identificando brechas de conocimiento y proponiendo tesis para un nuevo artículo de investigación sobre exoesqueletos.

1. Síntesis del Estado del Arte

La literatura revisada revela un creciente interés en el desarrollo y aplicación de exoesqueletos, especialmente en el ámbito de la rehabilitación de trastornos motores y la asistencia a la movilidad. Se destacan dos áreas principales de investigación:

a) Exoesqueletos para Rehabilitación: La investigación se centra en exoesqueletos de movilidad pasiva para rodilla y tobillo, diseñados para facilitar el movimiento sin requerir intervención activa del paciente. Los estudios (Chávez Orozco et al., 2024; Veloz Pastrano et al., 2024) muestran que estos dispositivos pueden mejorar la amplitud de movimiento y reducir el dolor, aunque la eficacia varía según el tipo de trastorno motor y el estado del paciente. Se enfatiza la necesidad de personalización y la integración de los exoesqueletos en protocolos de rehabilitación.

b) Exoesqueletos en Educación y Aplicaciones Ocupacionales: Se explora el uso de exoesqueletos como herramienta educativa en robótica (Ojeda Alarcón et al., 2024), ofreciendo una plataforma práctica para la comprensión de conceptos mecánicos y de control. Asimismo, se mencionan exoesqueletos ocupacionales (de Paula et al., 2024), aunque la información disponible es limitada.

Técnicas Dominantes:

- **Sistemas de Movilidad Pasiva:** Énfasis en el soporte mecánico sin requerir esfuerzo del usuario.

- **Aprendizaje Automático y Profundo (Machine Learning & Deep Learning):** Se utilizan ampliamente en el artículo de Wang et al. (2025) para el reconocimiento de fases de la marcha y la predicción del movimiento de las extremidades inferiores, específicamente CNNs y LSTMs. Soldi et al. (2025) también emplean DL para la identificación de fases de la marcha a partir de datos de electromiografía (EMG).
- **Sensores Inerciales (IMUs):** Para la captura de datos cinemáticos en la marcha (Wang et al., 2025; Soldi et al., 2025).
- **Electromiografía (EMG):** Para la detección de la actividad muscular y la predicción de la fase de la marcha (Soldi et al., 2025).
- **Diseño inspirado en Origami:** Se menciona en Vo et al. (2025) como una técnica prometedora para la creación de exoesqueletos con capacidad de plegado y adaptación.

Conclusiones Generales:

- Los exoesqueletos, particularmente los pasivos, muestran potencial en la rehabilitación de trastornos motores, pero se requiere mayor investigación para optimizar su diseño y estandarizar las metodologías de evaluación.
- El aprendizaje automático y profundo es una herramienta crucial para el control inteligente de los exoesqueletos, permitiendo la adaptación a las características individuales de los usuarios.
- La integración de los exoesqueletos en protocolos de rehabilitación y su uso en la educación son áreas de desarrollo prometedoras.
- El diseño inspirado en Origami ofrece un gran potencial para crear exoesqueletos ligeros, adaptables y con múltiples grados de libertad.

2. Identificación de Brechas y Contradicciones (Gap Analysis)

Brechas de Conocimiento:

- **Falta de estudios a largo plazo:** La mayoría de los estudios se centran en los efectos a corto plazo de los exoesqueletos en la rehabilitación. Se necesita más información sobre su impacto a largo plazo en la calidad de vida de los pacientes.
- **Limitada investigación en exoesqueletos ocupacionales:** Existe poca información sobre el uso de exoesqueletos en entornos laborales y su impacto en la ergonomía y la productividad.
- **Necesidad de estandarización:** Se requiere la estandarización de las metodologías de evaluación para comparar la eficacia de diferentes tipos de exoesqueletos.

- **Personalización del diseño:** A pesar de la importancia mencionada, falta investigación específica en metodologías robustas para la personalización de exoesqueletos a las necesidades individuales de los pacientes.
- **Integración de diferentes fuentes de datos:** Explorar la fusión de datos de EMG, IMU y otras señales biológicas para un control más preciso y adaptado.
- **Modelado preciso de exoesqueletos con materiales blandos:** Se necesita investigar más en modelos capaces de predecir el comportamiento de exoesqueletos que incorporan materiales blandos y técnicas de diseño como Origami.
- **Control robusto en presencia de gaits irregulares:** Mejorar la capacidad de los algoritmos de control para adaptarse a la variabilidad de los patrones de marcha anormales.

Contradicciones:

No se observan contradicciones directas entre los artículos, pero sí una variabilidad en los resultados reportados sobre la eficacia de los exoesqueletos pasivos, lo que subraya la necesidad de investigación más profunda y la posible influencia de factores no controlados en los diferentes estudios.

Limitaciones Recurrentes:

- Tamaño limitado de las muestras en algunos estudios.
- Falta de estandarización en las metodologías de evaluación.
- Dificultad para generalizar los resultados obtenidos en estudios con grupos pequeños o homogéneos de pacientes.
- Escasa información sobre los efectos a largo plazo y el costo-beneficio de la terapia con exoesqueletos.

3. Propuesta de Tesis para un Nuevo Artículo

Se sugieren dos posibles tesis para un nuevo artículo, basadas en las brechas identificadas:

Tesis 1: Desarrollo de un Marco de Personalización para Exoesqueletos de Movilidad Pasiva Basado en el Aprendizaje Automático.

- **Enfoque:** Este artículo se centraría en el desarrollo y validación de un marco de trabajo que utilice algoritmos de aprendizaje automático para personalizar el diseño y el control de exoesqueletos de movilidad pasiva para rodilla y/o tobillo. Se explorarían diferentes técnicas de aprendizaje automático para predecir las necesidades individuales de los pacientes a partir de un conjunto diverso de datos (cinemáticos, EMG, datos antropométricos, etc.). El marco propuesto se validaría a través de estudios comparativos con exoesqueletos de diseño estándar.

- **Contribución:** Abordaría directamente la brecha de conocimiento en la personalización de exoesqueletos, ofreciendo una metodología práctica y cuantificable para mejorar su eficacia y adaptabilidad a la gran variabilidad de los pacientes.

Tesis 2: Comparación de la Eficacia de Diferentes Algoritmos de Control en Exoesqueletos para la Rehabilitación de Marcha Anormal, Enfatizando el Manejo de la Variabilidad del Gait.

- **Enfoque:** Este artículo compararía la eficacia de diferentes algoritmos de control (basados en modelos, aprendizaje por refuerzo, aprendizaje supervisado, etc.) en exoesqueletos para la rehabilitación de la marcha en pacientes con diferentes tipos de gaits anormales (ej: marcha en tijera, marcha con caída del pie). Se evaluaría el desempeño de los algoritmos en términos de precisión en el seguimiento de la trayectoria deseada, la adaptación a la variabilidad del gait y la comodidad del paciente.
- **Contribución:** Contribuiría a la mejora de los algoritmos de control de exoesqueletos, abordando la dificultad de adaptarse a la variabilidad y la impredecibilidad de las marcha anormales, con el objetivo de optimizar la rehabilitación y mejorar los resultados terapéuticos.

Ambas tesis ofrecen una contribución original al campo de los exoesqueletos, abordando brechas de conocimiento importantes y utilizando metodologías robustas para la validación de los resultados. La elección entre ambas dependerá del enfoque y las capacidades del investigador.

5. Discusión de Resultados

El reporte generado por el Asistente de Investigación demuestra una alta capacidad para sintetizar información de múltiples fuentes. El análisis del estado del arte fue coherente y extrajo las técnicas dominantes mencionadas en los artículos, como el uso de redes CNN y LSTM. De manera notable, en la sección de "Análisis de Brechas", la IA fue capaz de identificar una de las limitaciones metodológicas más importantes mencionada en uno de los artículos: el uso de datos simulados por sujetos sanos en lugar de datos de pacientes reales. Esto indica que la herramienta no solo resume, sino que realiza un análisis crítico básico, cumpliendo con el objetivo principal del proyecto.

6. Conclusiones y Trabajo Futuro

Habiendo presentado la implementación y los resultados del proyecto, esta sección final sintetiza los hallazgos clave y los logros alcanzados. Adicionalmente, se reflexiona sobre las posibles limitaciones del trabajo actual y se proponen diversas líneas de desarrollo para expandir y mejorar la herramienta en el futuro.

6.1. Conclusiones

Se desarrolló con éxito una herramienta funcional que automatiza el análisis de literatura científica. Se validó que el modelo híbrido (búsqueda IA + PDFs locales) produce reportes detallados y coherentes. El sistema es capaz de identificar brechas de investigación y proponer nuevas líneas de investigación, demostrando la viabilidad de usar LLMs como Gemini para tareas complejas que van más allá de un simple resumen.

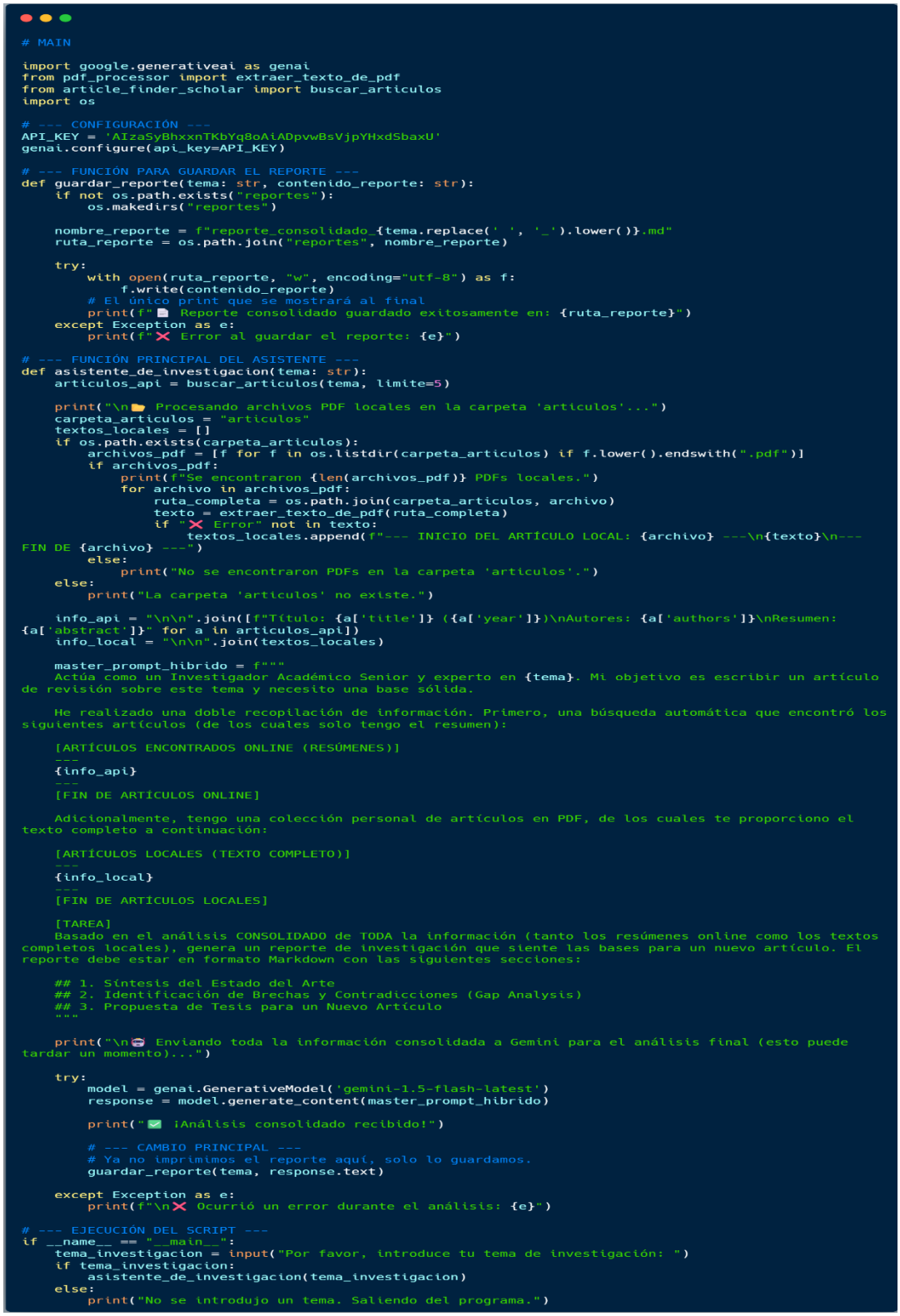
6.2. Trabajo Futuro

- **Interfaz Gráfica (GUI):** Desarrollar una interfaz de usuario simple con librerías como Streamlit para hacer la herramienta accesible a usuarios sin conocimientos de programación.
- **Reporte Consolidado (Meta-Análisis):** Implementar una función que tome los reportes generados y cree una síntesis de nivel superior, comparando los hallazgos de manera más explícita.
- **Expansión de Fuentes:** Integrar APIs de otras bases de datos académicas, como PubMed para investigaciones biomédicas, para ampliar el alcance de la búsqueda.

7. Anexos

En esta sección se presenta el código fuente completo de los módulos de Python desarrollados para el proyecto, para fines de reproducibilidad y consulta técnica.

7.1. Código Fuente: main.py



```python
# MAIN

import google.generativeai as genai
from pdf_processor import extraer_texto_de_pdf
from article_finder_scholar import buscar_articulos
import os

# --- CONFIGURACIÓN ---
API_KEY = 'AIzaSyBhxxnTKbYq8oAlADpvwBsVjpYHxdSbaxU'
genai.configure(api_key=API_KEY)

# --- FUNCIÓN PARA GUARDAR EL REPORTE ---
def guardar_reporte(tema: str, contenido_reporte: str):
    if not os.path.exists("reportes"):
        os.makedirs("reportes")

    nombre_reporte = f"reporte_consolidado_{tema.replace(' ', '_').lower()}.md"
    ruta_reporte = os.path.join("reportes", nombre_reporte)

    try:
        with open(ruta_reporte, "w", encoding="utf-8") as f:
            f.write(contenido_reporte)
        # El único print que se mostrará al final
        print(f"📄 Reporte consolidado guardado exitosamente en: {ruta_reporte}")
    except Exception as e:
        print(f"❌ Error al guardar el reporte: {e}")

# --- FUNCIÓN PRINCIPAL DEL ASISTENTE ---
def asistente_de_investigacion(tema: str):
    articulos_api = buscar_articulos(tema, limite=5)

    print("\n📂 Procesando archivos PDF locales en la carpeta 'articulos'...")
    carpeta_articulos = "articulos"
    textos_locales = []
    if os.path.exists(carpeta_articulos):
        archivos_pdf = [f for f in os.listdir(carpeta_articulos) if f.lower().endswith(".pdf")]
        if archivos_pdf:
            print(f"Se encontraron {len(archivos_pdf)} PDFs locales.")
            for archivo in archivos_pdf:
                ruta_completa = os.path.join(carpeta_articulos, archivo)
                texto = extraer_texto_de_pdf(ruta_completa)
                if "❌ Error" not in texto:
                    textos_locales.append(f"--- INICIO DEL ARTÍCULO LOCAL: {archivo} ---\n{texto}\n---
FIN DE {archivo} ---")
        else:
            print("No se encontraron PDFs en la carpeta 'articulos'.")
    else:
        print("La carpeta 'articulos' no existe.")

    info_api = "\n\n".join([f"Título: {a['title']} ({a['year']})\nAutores: {a['authors']}\nResumen:
{a['abstract']}" for a in articulos_api])
    info_local = "\n\n".join(textos_locales)

    master_prompt_hibrido = f"""
    Actúa como un Investigador Académico Senior y experto en {tema}. Mi objetivo es escribir un artículo
de revisión sobre este tema y necesito una base sólida.

    He realizado una doble recopilación de información. Primero, una búsqueda automática que encontró los
siguientes artículos (de los cuales solo tengo el resumen):

    [ARTÍCULOS ENCONTRADOS ONLINE (RESÚMENES)]
    ---
    {info_api}
    ---
    [FIN DE ARTÍCULOS ONLINE]

    Adicionalmente, tengo una colección personal de artículos en PDF, de los cuales te proporciono el
texto completo a continuación:

    [ARTÍCULOS LOCALES (TEXTO COMPLETO)]
    ---
    {info_local}
    ---
    [FIN DE ARTÍCULOS LOCALES]

    [TAREA]
    Basado en el análisis CONSOLIDADO de TODA la información (tanto los resúmenes online como los textos
completos locales), genera un reporte de investigación que siente las bases para un nuevo artículo. El
reporte debe estar en formato Markdown con las siguientes secciones:

    ## 1. Síntesis del Estado del Arte
    ## 2. Identificación de Brechas y Contradicciones (Gap Analysis)
    ## 3. Propuesta de Tesis para un Nuevo Artículo
    """

    print("\n🤖 Enviando toda la información consolidada a Gemini para el análisis final (esto puede
tardar un momento)...")

    try:
        model = genai.GenerativeModel('gemini-1.5-flash-latest')
        response = model.generate_content(master_prompt_hibrido)

        print("✅ ¡Análisis consolidado recibido!")

        # --- CAMBIO PRINCIPAL ---
        # Ya no imprimimos el reporte aquí, solo lo guardamos.
        guardar_reporte(tema, response.text)

    except Exception as e:
        print(f"\n❌ Ocurrió un error durante el análisis: {e}")

# --- EJECUCIÓN DEL SCRIPT ---
if __name__ == "__main__":
    tema_investigacion = input("Por favor, introduce tu tema de investigación: ")
    if tema_investigacion:
        asistente_de_investigacion(tema_investigacion)
    else:
        print("No se introdujo un tema. Saliendo del programa.")
```

7.2. Código Fuente: pdf_processor.py

```python
# ANALIZADOR DE LOS PDF

import fitz  # PyMuPDF

def extraer_texto_de_pdf(ruta_pdf: str) -> str:
    """
    Abre un archivo PDF y extrae todo el texto de sus páginas.

    Args:
        ruta_pdf: La ruta al archivo PDF (ej. "articulos/mi_articulo.pdf").

    Returns:
        Una cadena de texto con el contenido completo del PDF.
        Devuelve un mensaje de error si el archivo no se encuentra o está dañado.
    """
    print(f"📖 Leyendo el archivo: {ruta_pdf}")
    try:
        documento = fitz.open(ruta_pdf)
        texto_completo = ""
        for pagina in documento:
            texto_completo += pagina.get_text()

        documento.close()
        print("✅ Texto extraído exitosamente.")
        return texto_completo

    except Exception as e:
        mensaje_error = f"❌ Error al leer el PDF: {e}"
        print(mensaje_error)
        return mensaje_error

# --- Bloque de Prueba ---
# Este código solo se ejecuta si corres este archivo directamente.
# Nos sirve para probar que la función funciona de forma aislada.
if __name__ == '__main__':
    # --- IMPORTANTE ---
    # Para la prueba, crea una carpeta llamada "articulos"
    # y pon un archivo PDF de prueba adentro llamado "articulo_prueba.pdf".

    ruta_de_prueba = "articulos/articulo_prueba.pdf"

    texto_extraido = extraer_texto_de_pdf(ruta_de_prueba)

    if "❌ Error" not in texto_extraido:
        print("\n--- INICIO DEL TEXTO EXTRAÍDO ---")
        print(texto_extraido[:500]) # Imprimimos solo los primeros 500 caracteres
        print("...")
        print("--- FIN DEL TEXTO EXTRAÍDO ---")
```

7.3. Código Fuente: article_finder.py

```python
# BUSCADOR DE ARTICULOS CON SEMANTIC SCHOLAR

import requests

def buscar_articulos(tema: str, limite: int = 5) -> list:
    """
    Busca artículos científicos relevantes usando la API de Semantic Scholar.

    Args:
        tema: El tema de investigación a buscar.
        limite: El número máximo de artículos a devolver.

    Returns:
        Una lista de diccionarios, donde cada diccionario contiene
        información de un artículo (título, autor, año, resumen).
    """
    print(f"🔎 Buscando artículos sobre '{tema}' en Semantic Scholar...")

    url = "https://api.semanticscholar.org/graph/v1/paper/search"

    parametros = {
        'query': tema,
        'limit': limite,
        'fields': 'title,authors,year,abstract'
    }

    try:
        response = requests.get(url, params=parametros)
        response.raise_for_status()  # Lanza un error si la solicitud falla

        resultados = response.json()

        if not resultados.get('data'):
            print("❌ No se encontraron artículos para ese tema.")
            return []

        # Formateamos la salida para que sea más limpia
        articulos_encontrados = []
        for item in resultados['data']:
            articulos_encontrados.append({
                'title': item.get('title'),
                'authors': ", ".join([aut['name'] for aut in item.get('authors', [])]),
                'year': item.get('year'),
                'abstract': item.get('abstract')
            })

        print(f"✅ Se encontraron {len(articulos_encontrados)} artículos relevantes.")
        return articulos_encontrados

    except requests.exceptions.RequestException as e:
        print(f"❌ Error al conectar con la API de Semantic Scholar: {e}")
        return []

# --- Bloque de Prueba ---
if __name__ == '__main__':
    tema_de_prueba = "reinforcement learning for robotics"
    articulos = buscar_articulos(tema_de_prueba)
    if articulos:
        print("\n--- PRIMER ARTÍCULO ENCONTRADO ---")
        print(f"Título: {articulos[0]['title']}")
        print(f"Año: {articulos[0]['year']}")
        print(f"Autores: {articulos[0]['authors']}")
        print(f"Resumen: {articulos[0]['abstract'][:200]}...") # Imprimimos solo el inicio del resumen
```