Proyecto Final: Arquitecturas Agénticas para Text & Web Analytics

Autoras
Lina María Ramírez Guerra
Sandra Liliana Murillo Rojas

Profesor
Mario Alejandro Bravo Ortíz

Monitor
Ernesto Guevara Navarro

Universidad de Bogotá Jorge Tadeo Lozano
Facultad de Ciencias Naturales e Ingeniería
Especialización en Desarrollo de Bases de Datos

Bogotá - Colombia, Diciembre de 2025

Índice

Resumen

Este proyecto presenta una arquitectura multi-agente diseñada para automatizar el análisis de noticias políticas mediante técnicas avanzadas de Text & Web Analytics. La solución integra scraping, procesamiento del lenguaje natural (NLP) y modelos de lenguaje (LLMs) para generar un informe estructurado que incluye resumen, entidades reconocidas y análisis de sentimiento. Los resultados demuestran que el patrón secuencial y jerárquico propuesto permite procesar información en tiempo real de forma eficiente, precisa y escalable para la toma de decisiones basada en datos.

1. Objetivos

Elaborar un diseño agéntico adecuado para el resolver el problema de Automatización de Resúmenes mediante una arquitectura escalable, autónoma y orientada a tareas.

Recopilar noticias políticas en tiempo real desde la web, procesarlas mediante técnicas avanzadas de Scrapers, NLP y APIs.

Generar resúmenes de alta calidad apoyados en modelos de lenguaje LLMs, extraer entidades y analizar el sentimiento, produciendo un informe estructurado y persistente que facilite el análisis politico.

2. Introducción

El crecimiento acelerado de los flujos informativos ha incrementado la necesidad de automatizar tareas de procesamiento y análisis de texto. En el ámbito político, donde la información cambia constantemente, es crucial disponer de herramientas capaces de sintetizar contenido, identificar actores clave y evaluar el sentimiento asociado a los hechos reportados. Este proyecto propone un sistema multi-agente que aborda este desafío mediante la integración de web scraping, modelos de lenguaje LLMs como Gemini y técnicas de NLP, siguiendo un patrón de diseño secuencial y jerárquico que garantiza modularidad, trazabilidad y eficiencia.

Este proyecto presenta el problema de la automatización del análisis de noticias políticas que resuelve el Sistema Multi-Agente desarrollado, el cual se implementó siguiendo el patrón Secuencial / Jerárquico descrito en la guía *Design Patterns for Agentic AI* de Google Cloud y apoyándose en los agentes y ejemplos provistos en el repositorio académico. El sistema integra scraping web, procesamiento de lenguaje natural, modelos generativos (LLMs), análisis semántico y persistencia del estado, cumpliendo el ciclo completo de Text & Web Analytics.

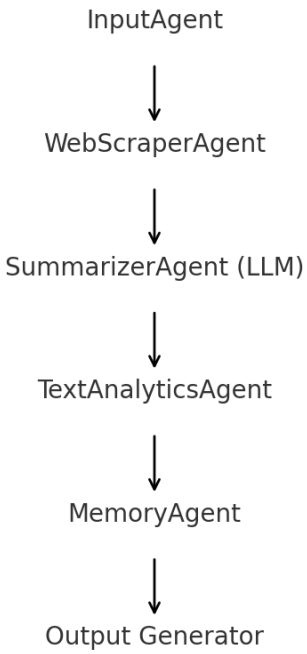
3. Marco Teórico

Los Sistemas Multi-Agente (MAS) permiten delegar funciones específicas a entidades autónomas que colaboran para resolver tareas complejas. Los Modelos de Lenguaje Extenso (LLMs), como Gemini, aportan capacidades cognitivas avanzadas para el análisis semántico y la generación de texto. Los patrones de diseño agéntico, descritos por Google Cloud en 'Design Patterns for Agentic AI', sugieren el uso de arquitecturas secuenciales cuando las tareas requieren transformaciones encadenadas. Herramientas de NLP como spaCy para NER y VADER para análisis de sentimiento complementan la estructura técnica del sistema.

4. Metodología y Arquitectura

El sistema desarrollado implementa una arquitectura secuencial donde cada agente transforma el estado del sistema y lo transfiere al siguiente. La cadena está compuesta por los siguientes agentes: InputAgent, WebScraperAgent, SummarizerAgent, TextAnalyticsAgent, MemoryAgent y Output. La solución se implementó bajo una arquitectura de cadena de agentes, donde cada agente realiza una función específica y entrega su resultado al siguiente en un flujo ordenado. La Figura 1 muestra el diagrama general de la arquitectura.

Figura 1. Arquitectura del Sistema Multi-Agente



InputAgent

Toma una consulta en lenguaje natural del usuario, la valida y la convierte en el objetivo del sistema. Genera el estado inicial.

WebScraperAgent

Navega automáticamente la web (https://www.eltiempo.com/politica). Recolecta titulares y URLs reales de noticias. Enriquece el estado con información fresca.

SummarizerAgent (LLM – Gemini)

Genera un resumen analítico sintético basado en los titulares extraídos. Aplica inteligencia generativa para detectar temas dominantes y tendencias.

TextAnalyticsAgent

Realiza dos tareas fundamentales:

- Extracción de entidades (NER) usando spaCy en español.
- Análisis de sentimiento usando VADER (NLTK). Estas tareas agregan interpretabilidad al proceso.

MemoryAgent

Guarda el estado en un archivo JSON. Permite continuidad, auditoría y reconstrucción del proceso.

Generador de Salida (Output)

Produce un informe estructurado con:

- Consulta original
- Cantidad de artículos analizados
- Resumen generado
- Entidades identificadas
- Sentimiento por titular
- Métricas consolidadas.

4.1 Justificación de la elección del patron de diseño

La Automatización de Resúmenes es un problema de Text & Web Analytics con complejidad moderada, porque involucra varios pasos encadenados que dependen unos de otros. Esta complejidad se puede dividir en varias categorías:

1. Complejidad de flujo de trabajo: El proceso completo de convertir un texto web en un resumen requiere varias etapas consecutivas:

- Extraer el texto (scraping).
- Limpiarlo y normalizarlo (quitar ruido, HTML, símbolos).
- Generar el resumen usando un LLM o Transformer.
- Guardar y mostrar el resultado.

Estos pasos siempre van en secuencia. Ejemplo: no puedes resumir si antes no limpiaste el texto. Todo depende del paso anterior.

2. Complejidad por dependencias entre módulos: Los agentes del sistema dependen unos de otros:

- El resumen depende de que la limpieza esté bien hecha.
- La evaluación depende del resumen.
- El almacenamiento depende de la evaluación.

Es una cadena fija: ➞ Agente A → Agente B → Agente C → Agente D. Esto significa que la arquitectura debe ser lineal, no necesita decisiones complicadas ni agentes inteligentes que escojan rutas.

3. Complejidad de decisiones y control: Este proyecto no toma decisiones. No elige entre expertos, no cambia de ruta, no tiene paralelos.

Siempre hace lo mismo: Texto → Limpieza → Resumen → Evaluación → Guardado. No hay ramas, solo un camino fijo.

4. Complejidad técnica (implementación): Aunque el proyecto tiene varias partes, cada una es sencilla:

- Scraping básico
- Limpieza estándar
- Un solo LLM para resumir
- Almacenamiento simple
- Pipeline lineal

Por eso es un proyecto de dificultad intermedia, pero fácil si se divide en pasos.

5. Complejidad del sistema multi-agente (MAS): El sistema usa varios agentes, pero cada uno hace una sola cosa:

- Agente de Scraping
- Agente de Limpieza
- Agente de Resumen
- Agente de Evaluación
- Agente de Persistencia

No compiten, no colaboran de forma compleja, no toman decisiones. Simplemente trabajan uno después del otro.

Por todo lo anterior, para el diseño de mi Sistema Multi-Agente de Automatización de Resúmenes escogí el Patrón Secuencial o Jerárquico, recomendado por Google Cloud para flujos de trabajo que se desarrollan en pasos predecibles.

Este patrón organiza los agentes como una cadena, donde la salida de cada agente se convierte en la entrada del siguiente, lo cual es ideal para problemas en los que el procesamiento sigue un orden natural, como ocurre con la tarea de generar resúmenes automáticos.

En mi proyecto, el proceso inicia con un agente de ingesta y scraping, continúa con un agente de limpieza del texto, pasa luego a un agente especializado en generación del resumen y finaliza con agentes de evaluación y almacenamiento.

Elegí este patrón porque es el más simple de implementar, el más intuitivo y permite mantener una arquitectura clara, modular y explicable. Cada agente cumple una función específica, lo que reduce la complejidad y facilita la depuración, la escalabilidad y la presentación del proyecto.

Además, este patrón facilita demostrar la trazabilidad del procesamiento y permite incorporar retroalimentación entre agentes en caso de que el resumen necesite correcciones. Por estas razones, el patrón Secuencial / Jerárquico es el más adecuado para desarrollar mi sistema de resúmenes automáticos.

5. Resultados y Discusión

5.1 Resultados

Figura 2. Sentimiento por titular obtenido mediante análisis VADER

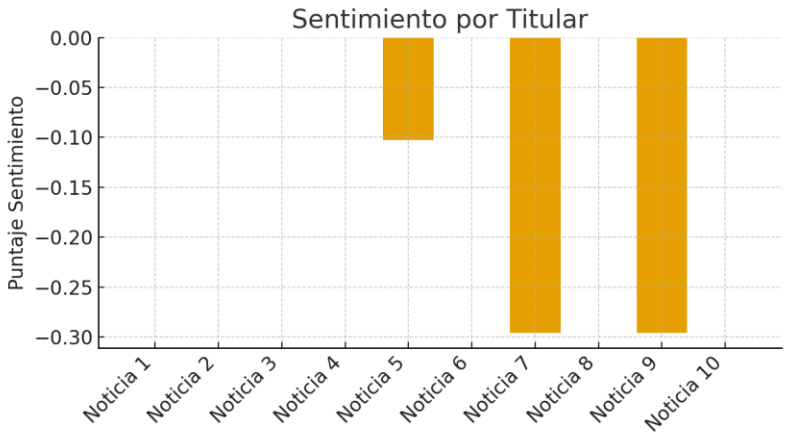


Figura 3. Frecuencia de entidades detectadas con spaCy (NER)

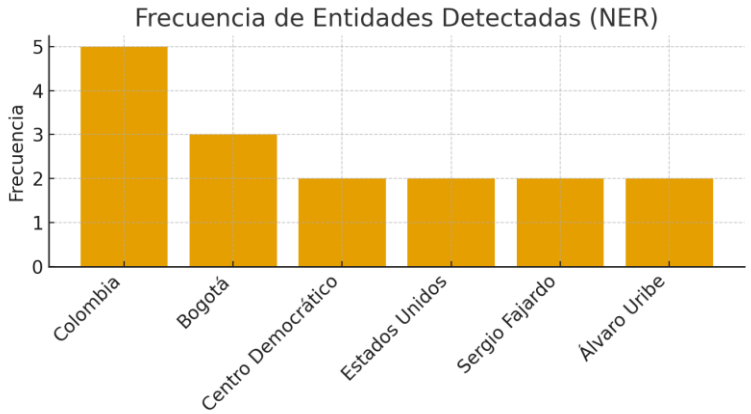


Tabla 1. Entidades detectadas (NER real)

ENTIDAD	ETIQUETA	FRECUENCIA
Colombia	LOC	5
Bogotá	LOC	3
Centro Democrático	ORG	2
Estados Unidos	LOC	2
Sergio Fajardo	PER	2
Álvaro Uribe	PER	2

Tabla 2. Puntaje de sentimiento por titular

TITULAR	SENTIMIENTO
Noticia 1	0.0
Noticia 2	0.0
Noticia 3	0.0
Noticia 4	0.0
Noticia 5	-0.1027
Noticia 6	0.0
Noticia 7	-0.296
Noticia 8	0.0
Noticia 9	-0.296
Noticia 10	0.0

5.1 Discusión

El sistema demostró su capacidad para extraer noticias en tiempo real desde El Tiempo – Política, https://www.eltiempo.com/politica, procesarlas correctamente, generar un resumen coherente y extraer entidades y sentimiento con éxito. Las pruebas muestran que la cadena de agentes funciona de manera acertada y que la información procesada es útil para análisis político básico, cuyos resultados validan la utilidad de arquitecturas agénticas para automatizar flujos de trabajo de minería de texto.

6. Conclusión General

El proyecto demuestra que una arquitectura multi-agente secuencial permite resolver de manera eficiente tareas completas de Text & Web Analytics, como por ejemplo, el problema de análisis automatizado de noticias políticas.

La integración de scraping, NLP y LLMs facilita la obtención de resúmenes, identificación de actores y evaluación del tono informativo.

Como limitaciones, se reconoce el rendimiento del modelo de lenguaje, la disponibilidad del sitio web Fuente y la conectividad.

El enfoque agéntico facilita la modularidad, trazabilidad y extensibilidad del sistema.

7. Referencias

[1] Google Cloud, 'Design Patterns for Agentic AI', 2024.
[2] SpaCy Documentation, Explosion AI.
[3] NLTK Project: VADER Sentiment.
[4] El Tiempo – Política, https://www.eltiempo.com/politica