



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Doble Grado en Ingeniería en Tecnologías de
Telecomunicación, y Análisis de Negocios

**Desarrollo de funcionalidades mediante inteligencia
artificial orientada al análisis de negocios.**

Autor
Alberto García Fábregas

Dirigido por
Luis Francisco Sánchez Merchante
Ricardo Ferrero

Madrid
Junio 2025

Alberto García Fábregas, declara bajo su responsabilidad, que el Proyecto con título **ClaudIA: Desarrollo de funcionalidades mediante inteligencia artificial orientada al análisis de negocios.** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2024/25 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Fecha: / /

Autoriza la entrega:

EL DIRECTOR DEL PROYECTO

Luis Francisco Sánchez Merchante

Fdo.: Fecha: / /

EL CO-DIRECTOR DEL PROYECTO

Ricardo Ferrero

Fdo.: Fecha: / /



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Doble Grado en Ingeniería en Tecnologías de
Telecomunicación, y Análisis de Negocios

**Desarrollo de funcionalidades mediante inteligencia
artificial orientada al análisis de negocios.**

Autor
Alberto García Fábregas

Dirigido por
Luis Francisco Sánchez Merchante
Ricardo Ferrero

Madrid
Junio 2025

Resumen

Desarrollo de funcionalidades mediante inteligencia artificial orientada al análisis de negocios

Autor: García Fábregas, Alberto.

Director: Sánchez Merchante, Luis Francisco.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas y Azvalor Asset Management.

El presente trabajo, titulado *Desarrollo de funcionalidades mediante inteligencia artificial orientada al análisis de negocios* implementa las técnicas más avanzadas de IA para optimizar procesos dentro del análisis empresarial. El proyecto pone a disposición de analistas y gestores una herramienta accesible y escalable que agilice la extracción de conocimiento de grandes volúmenes de información financiera y de mercado. Esta es una plataforma de análisis de negocios que combina Retrieval-Augmented Generation (RAG [4]) y agentes autónomos para extraer y sintetizar toda la información relevante.

Palabras clave: Agente, IA, RAG, LLM, análisis empresarial, NLP.

Introducción

El análisis de negocios moderno exige revisar informes, noticias y datos contables de forma rápida y precisa. Tradicionalmente, estas tareas se realizan de manera manual, consumiendo tiempo y presentando riesgo de omisión o errores de interpretación. Ante este panorama, surge la necesidad de automatizar la búsqueda y recuperación de información relevante.

Descripción del proyecto

Este proyecto responde a este reto incluyendo dos funcionalidades centrales:

- **Búsqueda local:** recuperación de fragmentos de texto dentro de documentos subidos por el usuario mediante un sistema RAG.

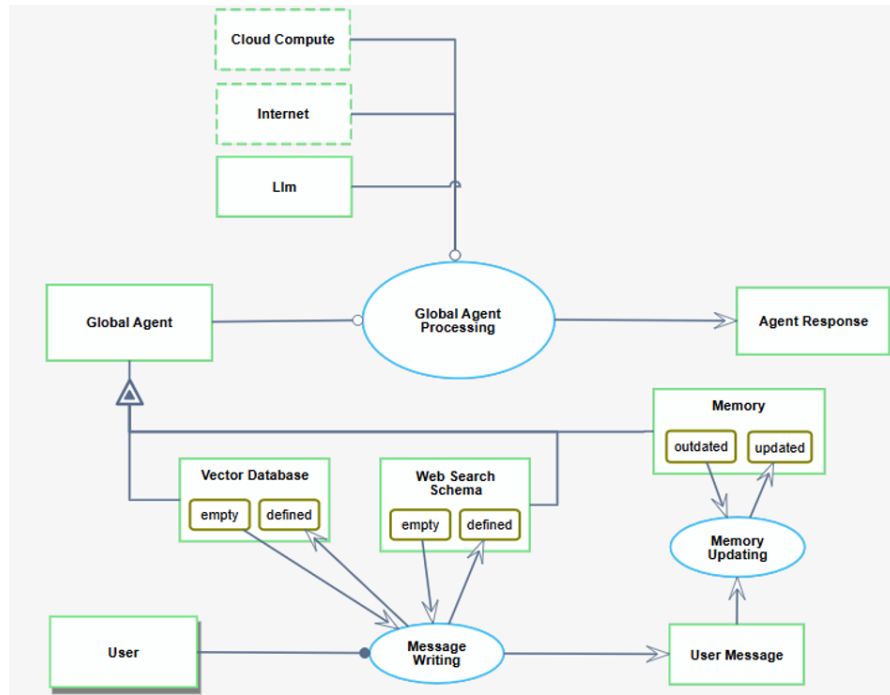


Figura 1: Diagrama OPM alto nivel del agente global

- **Búsqueda externa:** rastreo, filtrado y síntesis de noticias y recursos web, empleando agentes autónomos con motores de búsqueda que generan consultas, resumen y refinan iterativamente los resultados.

Se propone un **agente de inteligencia artificial** capaz de integrar ambas funcionalidades. Este agente presenta modelos de razonamiento y la capacidad de delegar tareas a otros agentes. Por otra parte, los analistas interactúan mediante una UI web, suben sus documentos y formulan consultas en lenguaje natural

Como queda descrito en el diagrama de la Figura 3, el agente tiene como núcleo un LLM, al que con el sistema diseñado (*Global Agent Processing*) y las entradas del usuario, responde generando un informe completo. Sobresale la flexibilidad de este agente global, puesto que puede trabajar con documentos adjuntados locales como también buscar en Internet para completar, e incluso hacer ambas tareas en paralelo.

Además, se exploran técnicas de vanguardia para optimizar cada etapa del proceso RAG; como el HyDE [2], Self-RAG [1], Adaptive [3] y muchas más, buscando optimizar el comportamiento de la IA y en concreto, de los agentes.

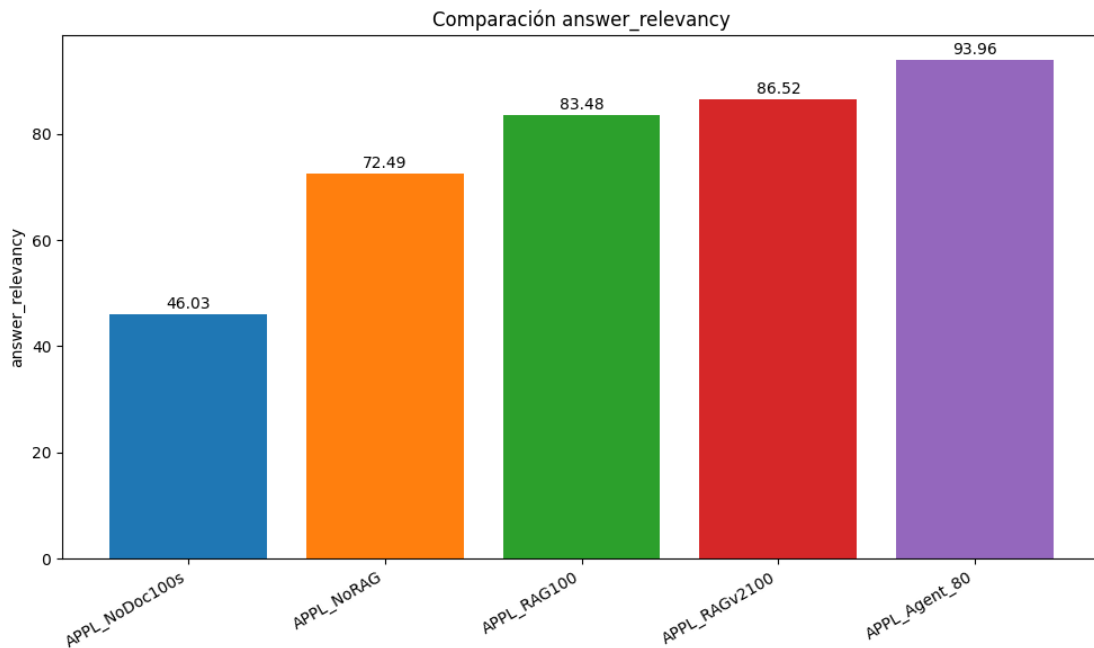


Figura 2: Comparativa entre los distintos sistemas de respuesta

Resultados

La arquitectura agencial demuestra en la funcionalidad local una mejora de 21 puntos porcentuales en el benchmark *FinRepQA* comparado con el sistema clásico de RAG (Figura 4). Además, el sistema es capaz de procesar tablas de PDFs con modelos de visión para su correcto uso en el análisis. Por otro lado, en la funcionalidad web, los agentes consiguen elaborar informes cuya aplicabilidad puede ir desde un screening de una empresa al estudio profundo de los directivos de una compañía.

Todos los agentes han sido desarrollados con trazabilidad, pudiendo saber en todo momento su estado actual. Esto permite, en tiempo real, conocer el progreso de la herramienta para cada petición.

Se han creado agentes semiautónomos que agilizan los procesos de análisis, eliminando tareas repetitivas. La funcionalidad RAG gestiona documentos locales, mientras que el buscador externo amplía la información en la red, evitando búsquedas manuales. El agente unificado integra ambas capacidades y simplifica su uso. La plataforma ofrece una interfaz intuitiva: el usuario sube archivos, define el formato de salida y lanza consultas en lenguaje natural, dejando el trabajo de extracción y síntesis en manos de la IA. Además, se permite establecer diálogos mediante lenguaje natural con estos agentes, gracias a la implementación de una

memoria conversacional.

Conclusiones

En conclusión, el proyecto aporta una herramienta práctica que libera tiempo de los analistas y eleva la calidad del análisis de negocios. Este trabajo demuestra la viabilidad de combinar RAG y agentes autónomos para transformar la forma en que las organizaciones extraen valor de los datos.

Summary

Development of functionalities through artificial intelligence oriented to business analysis

Author: García Fábregas, Alberto.

Supervisor: Sánchez Merchante, Luis Francisco.

Collaborating Entity: ICAI – Universidad Pontificia Comillas and Azvalor Asset Management.

This work, entitled *Development of functionalities through artificial intelligence oriented to business analysis*, implements the most advanced AI techniques to optimize processes within business analysis. The project provides analysts and managers with an accessible and scalable tool that speeds up the extraction of knowledge from large volumes of financial and market information. This is a business analysis platform that combines Retrieval-Augmented Generation (RAG [4]) and autonomous agents to extract and synthesize all relevant information.

Keywords: Agent, AI, RAG, LLM, business analysis, NLP.

Introduction

Modern business analysis demands reviewing reports, news and accounting data quickly and accurately. Traditionally, these tasks are carried out manually, consuming time and presenting a risk of omission or misinterpretation. Faced with this scenario, there arises the need to automate the search for and retrieval of relevant information.

Project Description

This project addresses this challenge by including two central functionalities:

- **Local search:** retrieval of text fragments within user-uploaded documents using a RAG system.

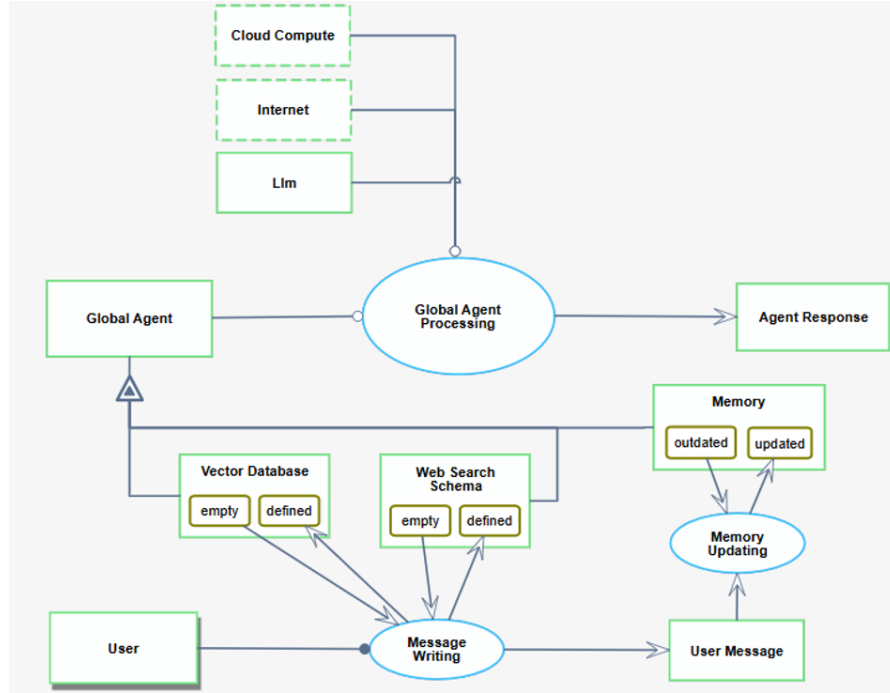


Figura 3: High-level OPM diagram of the global agent

- **External search:** crawling, filtering and synthesis of news and web resources, employing autonomous agents with search engines that generate queries, summarize and iteratively refine the results.

An **artificial intelligence agent** is proposed, capable of integrating both functionalities. This agent features reasoning models and the ability to delegate tasks to other agents. On the other hand, analysts interact via a web UI, upload their documents and pose queries in natural language.

As described in the diagram in Figure 3, the agent has an LLM at its core, to which, with the designed system (*Global Agent Processing*) and user inputs, it responds by generating a complete report. The flexibility of this global agent stands out, since it can work with locally attached documents as well as search the Internet to complete, or even perform both tasks in parallel.

Furthermore, cutting-edge techniques are explored to optimize each stage of the RAG process; such as HyDE [2], Self-RAG [1], Adaptive [3] and many more, aiming to optimize the behavior of the AI and, specifically, of the agents.

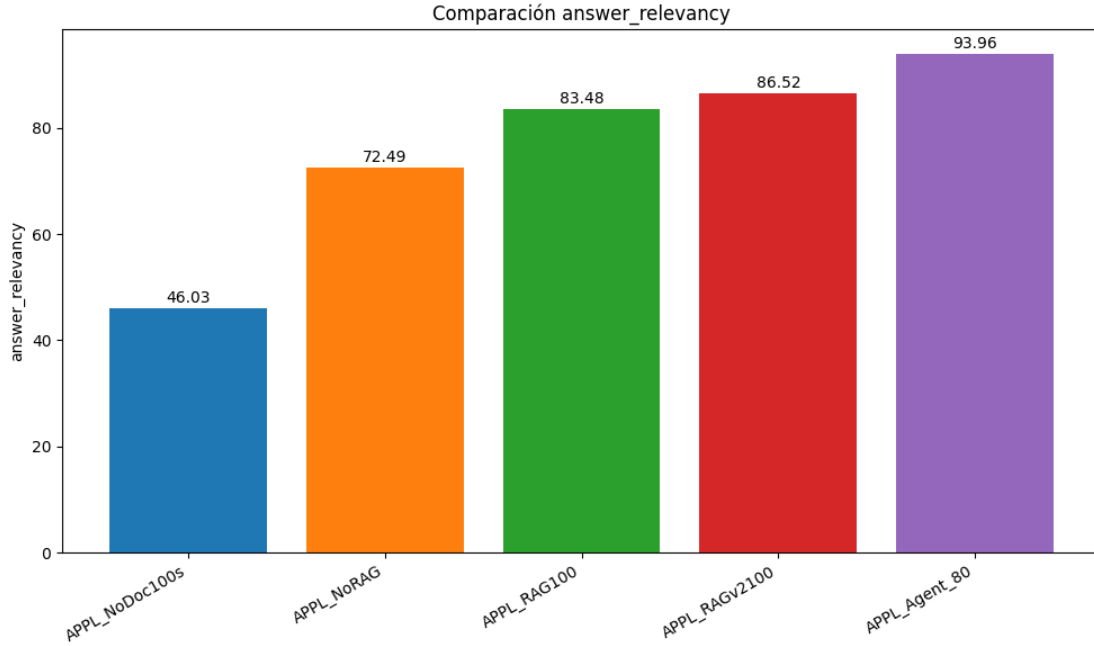


Figura 4: Comparison among the different response systems

Results

The agent-based architecture demonstrates, in the local functionality, an improvement of 21 percentage points on the *FinRepQA* benchmark compared to the classical RAG system (Figure 4). Moreover, the system is capable of processing tables from PDFs with vision models for their correct use in analysis. On the other hand, in the web functionality, the agents manage to produce reports whose applicability can range from a company screening to an in-depth study of a company's executives.

All agents have been developed with traceability, being able to know at all times their current state. This allows, in real time, to know the progress of the tool for each request.

Semi-autonomous agents have been created that speed up analysis processes by eliminating repetitive tasks. The RAG functionality manages local documents, while the external search engine expands the information on the web, avoiding manual searches. The unified agent integrates both capabilities and simplifies their use. The platform offers an intuitive interface: the user uploads files, defines the output format and submits queries in natural language, leaving the work of extraction and synthesis in the hands of the AI. Additionally, it is possible to establish dialogues in natural language with these agents, thanks to the implementation of

a conversational memory.

Conclusions

In conclusion, the project provides a practical tool that frees analysts' time and elevates the quality of business analysis. This work demonstrates the feasibility of combining RAG and autonomous agents to transform how organizations extract value from data.““

Abstract

El presente Trabajo de Fin de Grado aborda la creciente complejidad del análisis de empresas, donde los analistas financieros deben procesar volúmenes masivos de documentos (estados financieros, reglamentos internos, transcripciones de conferencias) de manera manual, lo que ralentiza la toma de decisiones y eleva el riesgo de omisiones. El objetivo principal consiste en diseñar e implementar un agente de inteligencia artificial que automatice las tareas repetitivas de búsqueda y síntesis de información, tanto en fuentes locales (documentos cargados por el usuario) como externas (noticias y recursos web), reduciendo la fricción y aumentando la productividad.

Para ello, se trabajó con herramientas como Langchain y Langgraph para la orquestación de pipelines de IA, junto con la modelización OPM para documentar arquitecturas y flujos de trabajo.. En la funcionalidad de búsqueda local se implementó un agente RAG con metadatos enriquecidos para extraer y vectorizar fragmentos de texto, optimizando la recuperación semántica y minimizando alucinaciones. Paralelamente, se desarrollaron agentes de búsqueda externa, incorporando refinamientos en generación de consultas, verificación iterativa y memoria conversacional con una arquitectura agencial.

Los resultados experimentales revelan un aumento de aproximadamente 11 puntos porcentuales en la precisión de las respuestas con respecto a un RAG básico, gracias a los bucles de validación del agente, así como una mejora notable en la calidad de informes. La herramienta fue validada con analistas de Azvalor Asset Management, quienes destacaron la autonomía controlada del agente, la interfaz conversacional y la capacidad para generar informes estructurados.

En conclusión, el prototipo desarrollado cumple los objetivos planteados, ofreciendo una solución escalable y fácilmente integrable en entornos profesionales, sentando las bases para su despliegue en producción, la incorporación de guardrails de auditoría y futuras extensiones funcionales.

Referencias del Resumen y Abstract

- [1] Akari Asai et al. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. 2023. arXiv: 2310.11511 [cs.CL]. URL: <https://arxiv.org/abs/2310.11511>.
- [2] Luyu Gao et al. *Precise Zero-Shot Dense Retrieval without Relevance Labels*. 2022. arXiv: 2212.10496 [cs.IR]. URL: <https://arxiv.org/abs/2212.10496>.
- [3] Soyeong Jeong et al. *Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity*. 2024. arXiv: 2403.14403 [cs.CL]. URL: <https://arxiv.org/abs/2403.14403>.
- [4] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". En: *Advances in Neural Information Processing Systems* 33 (2020), págs. 9459-9474.

*A mis padres,
por las oportunidades que me han brindado.*

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mis directores de TFG, cuya guía ha sido fundamental para el desarrollo de este trabajo. En primer lugar, agradezco a Luis Francisco Sánchez Merchante su cercanía, su inquebrantable apoyo técnico y su habilidad para conferir al proyecto la rigurosidad metodológica necesaria. Su disposición constante a resolver dudas y afinar cada detalle ha elevado la calidad de este estudio. En segundo lugar, mi gratitud a Ricardo Ferrero, cuya profesionalidad y visión estratégica más allá de los aspectos puramente tecnológicos me han servido de verdadero mentor en cada fase. Su generosidad al compartir conocimientos y su orientación de alto nivel han sido cruciales para dar sentido y relevancia a los resultados obtenidos.

Quisiera también reconocer a Azvalor Asset Management, y muy especialmente a Carlos Romero, por facilitar esta valiosa colaboración y brindarme acceso a recursos y datos de primer orden. A Javier Campos, Pedro Ordóñez y Diego Costa, les agradezco su tiempo, su apoyo constante y la cercanía con que han acompañado cada iteración del proyecto. Sin su feedback práctico y sus sugerencias, el enfoque aplicado de este trabajo no habría sido posible.

Finalmente, agradezco a mi familia y amigos por su paciencia, ánimo y comprensión durante las largas jornadas de dedicación; y a todas las personas que, de una forma u otra, han contribuido al éxito de este TFG. Su confianza y estímulo han sido el motor que me ha impulsado a avanzar hasta la meta.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	3
1.3. Objetivos	4
1.4. Metodología	5
2. Estado del Arte	7
3. Fundamentos Teóricos	13
3.1. LLM	13
3.1.1. Embeddings	14
3.1.2. Otras características de los LLM	14
3.1.3. Modelos via API	14
3.2. RAG	15
3.3. Prompt Engineering	17
3.3.1. Modelos de razonamiento	18
4. Funcionalidad de Búsqueda Local	21
4.1. RAG con metadatos enriquecidos	21
4.1.1. Objetivos de la Prueba de Concepto	22
4.1.2. Descripción del Workflow	22
4.1.3. Especificaciones técnicas	23
4.1.4. Resultados	24
4.1.5. Aspectos a mejorar	26
4.1.6. Conclusión	26
4.1.7. Problemática en el procesamiento de documentos	27
4.1.8. Implementación actual	27
4.1.9. Soluciones implementadas	28
4.1.10. Solución final	30
4.2. Otras técnicas implementadas	32
4.2.1. Chunking	32

4.2.2.	Hybrid Search	33
4.3.	RAG Agent	35
4.3.1.	Objetivos	35
4.3.2.	Descripción de la Arquitectura	36
4.3.3.	Especificaciones Técnicas	37
4.3.4.	Resultados	38
4.3.5.	Conclusión	38
5.	Funcionalidad Búsqueda	41
5.0.1.	Tavily Seach API	41
5.1.	Agente de Búsqueda 1.0	41
5.1.1.	Arquitectura y funcionamiento	42
5.1.2.	Especificaciones	43
5.1.3.	Resultados	44
5.1.4.	Conclusiones	44
5.2.	Agente de Búsqueda 2.0	45
5.2.1.	Arquitectura y funcionamiento	45
5.2.2.	Especificaciones	47
5.2.3.	Resultados	48
5.2.4.	Conclusión	49
5.3.	Global Agent	51
5.3.1.	Objetivos del Agente Global	51
5.3.2.	Arquitectura y Funcionamiento	52
5.3.3.	Mejoras de los agentes anteriores	54
5.3.4.	Especificaciones Técnicas	56
5.3.5.	Resultados	56
5.3.6.	Conclusiones	56
6.	Resultados	59
6.1.	Testing RAG	59
6.1.1.	Metodología para la evaluación de agentes	59
6.1.2.	Número óptimo de chunks	61
6.1.3.	Comparación de la precisión de los distintos sistemas y evoluciones	62
6.1.4.	Estrategias de generación de consultas y recuperación de fragmentos	64
6.1.5.	Agente RAG	66
6.1.6.	Conclusión	67

7. Discusión	69
7.1. Autonomía de los agentes	69
7.2. El lugar de la IA en análisis de empresas	69
7.3. Ética y trazabilidad	70
8. Conclusión	71
8.1. Objetivos cumplidos	71
8.2. Necesidad real	72
8.3. Siguiendo pasos: producción	72
8.4. Proyecto de ingeniería	72
9. Líneas futuras	75
9.1. Optimización avanzada de los agentes	75
9.2. Despliegue y estrategia <i>DevOps</i>	76
9.3. Evolución de los modelos de inteligencia artificial	76
9.4. Protocolos de colaboración entre agentes	76
9.5. Estándares de integración de herramientas MCP	76
9.6. Memoria a largo plazo	77
9.7. Experiencia de usuario y evaluación de la calidad	77
Anexos	79
A. Alineación de los Objetivos de desarrollo Sostenible (ODS)	79
B. Código Proyecto	81
C. Vídeos Resultados	83
D. Diagramas OPM y OPL	85
D.1. Agente RAG	85
D.2. Agente de Búsqueda Web	87
D.3. Agente Global	88
Bibliografía	93

Índice de figuras

1.	Diagrama OPM alto nivel del agente global	IV
2.	Comparativa entre los distintos sistemas de respuesta	V
3.	High-level OPM diagram of the global agent	VIII
4.	Comparison among the different response systems	IX
1.1.	Evolución de los bonos, letras del tesoro y acciones de 1926 a 2024.[37]	3
1.2.	Evolución de los bonos, letras del tesoro y acciones de 1926 a 2024.[26]	4
2.1.	Comparativa de funcionalidades	10
4.1.	Diagrama de flujo de RAG con metadatos enriquecidos.	23
4.2.	Respuesta: Análisis Consejero Delegado.	25
4.3.	Tabla original usada para la comparativa.	27
4.4.	Tabla procesada con PyPDF2 [20]	28
4.5.	Tabla procesada con pdfplumber [45]	28
4.6.	Tabla procesada con camelot [9]	29
4.7.	Tabla procesada con tabula [3]	29
4.8.	Comparativa de otra tabla procesada con LlamaParse [32] en modo estándar (formato <i>markdown</i>)	31
4.9.	Tabla procesada con LlamaParse [32] en modo avanzado (formato <i>markdown</i>)	32
4.10.	Esquema OPM: Arquitectura del Agente RAG	36
4.11.	Interfaz del Agente RAG	38
5.1.	Respuesta de ChatGPT Search a petición compleja.	42
5.2.	Diagrama de flujo del Agente de Búsqueda 1.0	43
5.3.	Respuesta del Agente de Búsqueda 1.0	44
5.4.	Diagrama de flujo del Agente de Búsqueda 2.0	46
5.5.	Interfaz del Agente de Búsqueda Web	49
5.6.	Esquema OPM: Arquitectura del Agente Global	52
5.7.	Esquema OPM: Estructura Interna del <i>Global Agent Processing</i> 5.6	53
5.8.	Interfaz de usuario del agente global	55

6.1.	Evolución de resultados según el número de chunks	62
6.2.	Comparativa entre los distintos sistemas de respuesta	63
6.3.	Relevancia de respuesta para cada configuración.	64
6.4.	Comparación de puntuación global (Overall Score) para diferentes métodos de recuperación de contexto.	65
6.5.	Comparación de rendimiento general entre Query Expansion y HyDE.	66
6.6.	Comparativa entre agente y query expansion	67
8.1.	Comparativa de la herramienta desarrollada con el mercado	71
D.1.	OPM del Agente RAG	86
D.2.	OPM Agente de Búsqueda Web	87
D.3.	Diagrama OPM Agente Global	88
D.4.	Diagrama OPM Agente Global Processing	90

Acrónimos

<i>ICAI</i>	Instituto Católico de Artes e Industrias
<i>PFC</i>	Proyecto Fin de Carrera
<i>IA</i>	Inteligencia Artificial
<i>RNN</i>	Redes Neuronales Recurrentes
<i>LSTM</i>	Long–Short Term Memory
<i>DNN</i>	Deep Neural Networks
<i>LLM</i>	Large Language Models
<i>LVM</i>	Large Vision Models
<i>NLP</i>	Natural Language Processing
<i>BM25</i>	Búsqueda por palabras clave
<i>RAG</i>	Retrieval-Augmented Generation
<i>API</i>	Interfaz de Programación de Aplicaciones
<i>PE</i>	Positional Encoding
<i>GPU</i>	Graphics Processing Unit
<i>ODS</i>	Objetivos de Desarrollo Sostenible
<i>LSEG</i>	London Stock Exchange Group
<i>OPL</i>	Object-Process Language

Capítulo 1

Introducción

1.1. Contexto

A lo largo de las últimas décadas, el mundo de la inteligencia artificial (IA) ha evolucionado drásticamente. Desde modelos simbólicos y basados en reglas, a modelos estadísticos de aprendizaje profundo. A principios de los años 90, las redes neuronales recurrentes (RNN) empezaban a demostrar la viabilidad de capturar relaciones temporales en series de datos mediante interconexiones recurrentes.

Más adelante, fue la aparición de las Long-Short Term Memory (LSTM) [23], la arquitectura especializada de las RNN que permitió mantener esas dependencias a largo plazo. Con esto, se fue avanzando hacia las redes neuronales profundas (DNN o Deep Neural Networks), que aumentaban las capacidades al aumentar la profundidad de las redes, todo ello gracias a los avances del hardware.

Finalmente, en 2017 se presenta la arquitectura de los transformers [48], el punto de inflexión que estableció el inicio de la tecnología IA más popular actualmente. Mediante su mecanismo de atención, los transformers permiten capturar dependencias globales con mayor eficiencia. Esta publicación desbloqueó el camino hacia los modelos de lenguaje de gran escala que tanto se han popularizado en los últimos años, conocidos también como LLM o Large Language Models en inglés.

A partir de aquí, modelos como BERT [14], GPT-3 [7] y otros han proporcionado resultados fascinantes, siendo capaces de entablar conversaciones, respondiendo preguntas y hasta realizando problemas matemáticos. Estos modelos de lenguaje requieren de entrenamientos largos y computacionalmente intensivos, necesitando cantidades de información que superan la cantidad que un humano puede llegar a procesar en toda su vida.

Además, estos LLM dependen mucho de la petición del usuario, conocido como prompt. Los LLM carecen de contexto más allá de lo proporcionado en estos prompts, es por ello que el rendimiento de los mismos es radicalmente afectado por

el prompt del usuario. A raíz de esto, surge el "Prompt Engineering", consistente en el estudio y perfeccionamiento de los prompts para maximizar la precisión en la respuesta de los modelos.

Una técnica clave dentro del ámbito del Prompt Engineering es la de la Cadena de Pensamiento o Chain of Thought (CoT) [50] en inglés. Esta técnica aumenta cerca de un 40 % en precisión [49] debido a que el modelo toma unas etapas intermedias de razonamiento que le llevan a una respuesta más acertada que si directamente la intentase responder. En otras palabras, pedir de forma activa que los modelos hagan las etapas intermedias aporta una mayor probabilidad de que la respuesta sea acertada. En consecuencia, del concepto de CoT surgen los ahora llamados modelos de razonamiento. Estos modelos son entrenados para generar esta cadena de pensamiento y son afinados para tener una cadena de pensamiento optimizada. Estos modelos son los más novedosos y en este proyecto se estudia cómo se pueden emplear.

Por otra parte, la inversión desempeña un papel esencial en la creación de valor y el crecimiento económico. Los analistas financieros son clave en este proceso, ya que su labor consiste en evaluar oportunidades, medir riesgos y tomar decisiones informadas basadas en el análisis detallado de datos empresariales y del mercado. Como dice la frase: "No se puede medir lo que no se ve", lo que resalta la importancia de contar con herramientas capaces de procesar y hacer visibles grandes volúmenes de información. La reciente cantidad de información y la complejidad de los mercados representan un desafío para mantener la precisión y eficiencia en sus evaluaciones.

Concretamente, en el análisis de empresas, muchos analistas emplean horas estudiando extensos documentos publicados con cuentas anuales, reglamentos internos, transcripciones de conferencias, etc, todo ello agregando centenas de páginas de información y decenas de tablas de contabilidad que analizar por empresa. Cuando la cantidad de información incrementa a dicho nivel el análisis se convierte en una cuestión de tiempo, cuanto más información se quiere tener en cuenta, más tiempo se debe dedicar. Por ende, tecnologías como la inteligencia artificial, capaces de resumir información o extraerla, pueden ser muy útiles en este proceso.

La inversión tiene múltiples variantes y alternativas, de las más destacadas son la inversión en letras del tesoro, en bonos o en acciones. La Figura 1.1 demuestra como la inversión en acciones (renta variable) es más rentable, gráfico actualizado por New York Life [37], originalmente publicado en "Stocks, Bonds, Bills, and Inflation Historical Returns (1926-1987)"1.1. No obstante, es más arriesgada que otras opciones, pero, al menos mirando al pasado, es la opción más rentable a largo plazo. Dentro de la inversión en acciones, se distinguen principalmente dos tipos: las acciones de crecimiento, que son aquellas de empresas en sus primeros años que están creciendo en el mercado, y las acciones en valor, donde destacan

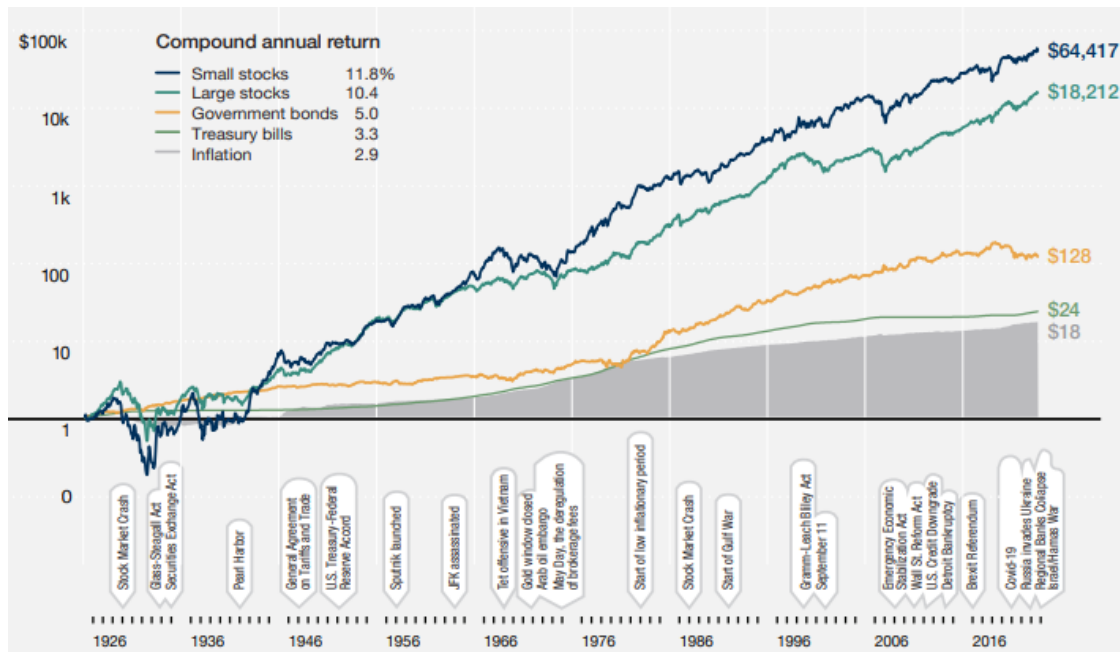


Figura 1.1: Evolución de los bonos, letras del tesoro y acciones de 1926 a 2024.[37]

empresas ya establecidas que generan flujos de caja positivos de forma recurrente y suelen tener poca deuda. A raíz de esto, ciertos inversores se dedica enteramente un estilo, a otro o a una mezcla. Históricamente, el value investing o inversión en valor es un 4.4 % más rentable, como se observa en la Figura 1.2, estudio realizado por Dimensional Hong Kong [26]. Es importante recalcar que el estudio del pasado no garantiza el desempeño futuro.

Entonces, el trabajo realizado a continuación se centra en procesos basados en el "value investing". Concretamente, en aquellos que más información es necesaria, esto es cuando se trabajan los documentos de la empresa como cuentas anuales, reglamentos internos, ... En este ámbito la IA puede aportar mucho puesto que su principales virtudes son las de resumen de texto y extracción de información. Sin embargo, en el capítulo del estado del arte podemos ver cómo este trabajo pretende ir más allá de las soluciones existentes.

1.2. Motivación

El análisis de empresas es un proceso crítico en la toma de decisiones estratégicas, especialmente en inversiones. A pesar de los avances logrados con herramientas como FactSet [19] y Bloomberg [6], que facilitan la recopilación y el análisis de datos, el proceso sigue siendo demandante en términos de tiempo debido a tareas

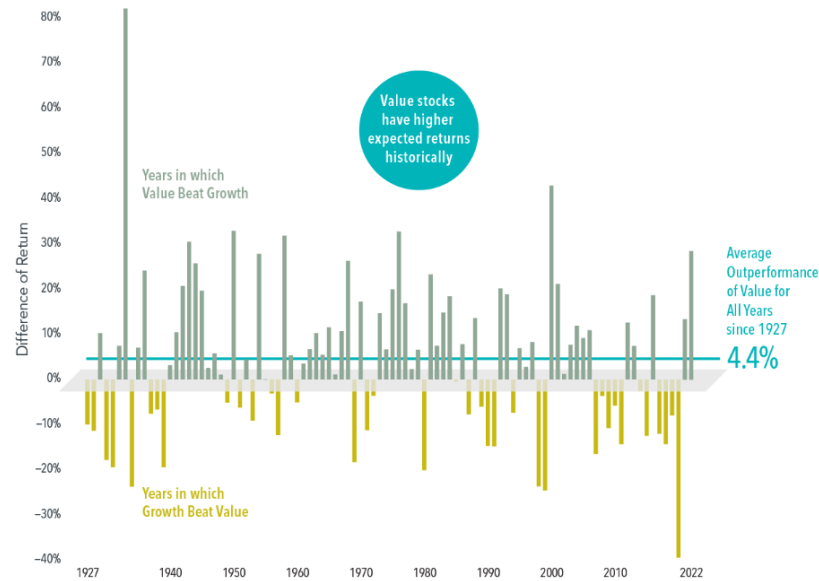


Figura 1.2: Evolución de los bonos, letras del tesoro y acciones de 1926 a 2024.[26]

repetitivas y a la fragmentación de información. Aunque soluciones basadas en inteligencia artificial, como las ofrecidas por RavenPack [40], Alphasense [1], FinChat [21] y LSEG [33] han optimizado aspectos específicos como la gestión del conocimiento y el análisis de noticias, no abordan de manera integral las necesidades del análisis empresarial.

Es por ello que este trabajo propone desarrollar un agente de inteligencia artificial que actúe como un compañero virtual para los analistas, integrándose en su flujo de trabajo para facilitar la recopilación, organización y presentación de datos clave de las empresas. Este agente no solo automatizará tareas repetitivas, sino que también proporcionará informes y comparativas, mejorando la eficiencia y la precisión del análisis. Al hacerlo, se busca no solo optimizar los procesos actuales, sino también establecer un nuevo estándar en el uso de la IA aplicada al análisis de empresas.

1.3. Objetivos

El propósito entonces es el de desarrollar un programa que mejore la productividad de los analistas, realizando tareas repetitivas por ellos, como la investigación en múltiples documentos o en la web. Es decir, este trabajo se centrará en dos funcionalidades:

- Búsqueda local: el agente será capaz de extraer la información necesaria de

los documentos que se suban a la plataforma.

- Búsqueda externa: el agente podrá buscar en noticias y recursos en la web para obtener información sobre la petición del usuario.

Como principal objetivo se busca desarrollar una herramienta en forma de un agente IA que facilite información relevante para el análisis de un negocio. Por un lado, en el análisis externo, los usuarios podrán incluir un esquema que quieran completar, ya sea de información de una empresa, de un aspecto de la misma o de una persona de dicha empresa. Por otra parte, en la búsqueda local, los usuarios podrán cargar en el momento documentos para extraer información y generar informes de los mismos.

Para reducir la fricción, se desarrollará una interfaz de chatbot que permita interactuar con el agente. De esta manera, se podrá usar el agente a través de lenguaje natural. Además, dicha interacción permitirá establecer una conversación en la cual el agente tendrá una memoria para cada conversación.

1.4. Metodología

Para llevar a cabo este trabajo de investigación, se adoptará una metodología de desarrollo ágil, similar a Scrum [43]. El proyecto se dividirá en sprints de dos a tres semanas de duración, al término de los cuales se realizarán reuniones de seguimiento para evaluar los avances, identificar posibles obstáculos y ajustar el alcance del proyecto si fuera necesario. La plataforma Teams será utilizada como herramienta de colaboración principal, facilitando la comunicación con los tutores y el seguimiento del progreso del proyecto. Además, Teams servirá como repositorio de toda la documentación generada, desde la planificación inicial hasta los resultados finales, garantizando así la trazabilidad y la transparencia del proceso.

El desarrollo del agente de inteligencia artificial se enfocará en la implementación de un prototipo funcional que permita interactuar con datos empresariales, ofreciendo resultados concretos y aplicables. Este enfoque práctico garantizará que el producto final sea una herramienta útil y tangible para analistas, más allá de una conceptualización teórica. Un riesgo que se buscará mitigar es el de las alucinaciones, intentando evitar estos errores en la medida de lo posible.

Esta programación de agentes se realizará con las librerías de Langchain [10] y Langgraph [29] principalmente. El ecosistema de Langchain está en constante crecimiento, es básicamente una librería que facilita las llamadas a modelos y la concatenación de las mismas. Lo que son aproximadamente 20 líneas de código para llamar a un API de un proveedor de modelos de IA y extraer su respuesta, se traduce en 4 con esta librería de código. Además, Langgraph es el siguiente paso, facilita la programación de agentes. No llega a ser un framework de agentes, pero

sí permite crear agentes basados en grafos que siguen ciertos flujos de trabajo. A esto se le añaden todas las integraciones de Langchain que le permiten conectar de una forma muy sencilla herramientas externas para que los agentes interactúen con ellas.

Se elige este ecosistema de librerías por su versatilidad, su gran catálogo de integraciones junto con su capacidad de parametrización y personalización. Sin embargo, la curva de aprendizaje es bastante más grande que otros frameworks. Empresas como Crewai [35] proporcionan un framework de alto nivel, fácil de entender e implementar, pero no permite diseñar los agentes a más bajo nivel, reduciendo el nivel de personalización. También, a lo largo del desarrollo de este proyecto surgieron otros frameworks como SmolAgents [25] y AutoGen [44] que, aparte de estar en una fase muy temprana de desarrollo, también carecen de profundidad en la personalización al ser herramientas de más alto nivel. Por último, la herramienta n8n [36] es de las más populares para crear flujos de trabajo con IA o agentes. Sin embargo, también es una herramienta no-code (sin obligación de usar código) y no permite la modificación a bajo nivel de los agentes. Aún así, todas estas herramientas son muy buenas para prototipados rápidos y despliegue sencillos con grandes capacidades de integrar con múltiples herramientas externas. Por ello, el ecosistema de Langchain es perfecto para este proyecto puesto que se busca ir en profundidad a personalizar cada pipeline, como por ejemplo el proceso de RAG [30] que se explora en el capítulo de la funcionalidad local.

Para facilitar el diseño y posterior explicación de los desarrollos, agentes y procesos, se emplea la metodología OPM: Object Process Modeling [15]. Esta metodología permite exponer diagramas de flujo de cualquier tipo de contexto de una forma estandarizada. Es decir, permite entender sistemas complejos mediante objetos, procesos y enlaces definidos. Esta metodología presenta los procesos tanto visualmente como de forma escrita, con el Object-Process Language (OPL), el cual está constituido por oraciones que definen los objetos, los procesos y sus relaciones. Estos flujos se desarrollan con el sandbox de la herramienta de OPCloud [16], el cual genera el OPL de forma automática en base al diagrama elaborado. Con esta metodología se facilitará el entendimiento y presentación de los agentes y sus flujos de trabajo con una metodología estandarizada bajo la ISO 19450:2024.

Durante el desarrollo del proyecto, para obtener un enfoque práctico y real, el proyecto será realizado en colaboración con Azvalor Asset Management [5], para evaluar y validar la herramienta con la retroalimentación de analistas financieros con años de experiencia, garantizando su utilidad y eficacia en escenarios reales. A lo largo del año se fueron realizando reuniones periódicas para determinar los siguientes pasos y funcionalidades de los agentes.

Capítulo 2

Estado del Arte

En el sector de la industria que presta servicios o desarrolla productos de IA para empresas está en continua evolución; principalmente impulsado por el reciente auge de la IA generativa cuyos modelos se mejoran mes a mes. Este incremento en el rendimiento de los modelos se traslada a todos los sectores y abre la puerta a la creación de nuevos productos basados o a la mejora de herramientas existentes. Esta revisión del estado del arte, centrará el foco de atención en las soluciones existentes que dan soporte en el sector financiero, concretamente los servicios que ofrecen apoyo en el análisis empresarial.

A continuación, se presentan cuatro soluciones destacadas: RavenPack (News Analytics) [40], LSEG (Machine Readable News) [33], AlphaSense [1] y FinChat.io [21]. Con un análisis individual de sus funcionalidades, casos de uso y principales características.

RavenPack News Analysis

En primer lugar, el producto News Analytics de RavenPack [40] es una herramienta centrada en el análisis de noticias con IA. Ofrece datos estructurados en tiempo real obtenidos de noticias y redes sociales. Entre sus principales funcionalidades destaca el análisis de sentimiento, proporcionando una puntuación positiva o negativa a cada noticia. Este análisis viene depurado tras muchos años de experiencia, llegando a desarrollar un sistema que combina modelos avanzados de machine learning con algoritmos basados en lenguas lingüísticas predefinidas. News Analytics proporciona un etiquetado semántico, que añade metadatos a las noticias para poder identificar y posteriormente filtrar en base a temas, compañías y eventos. El sistema reconoce entidades globales y clasifica eventos según su taxonomía para crear una knowledge graph. Según su documentación, este servicio de Ravenpack procesa información de fuentes premium, consistentes en periódicos autorizados a nivel nacional, regional y local, además de otros softwares como

Factset. Por último, a partir de las noticias, esta herramienta obtiene métricas avanzadas. Indicadores que facilitan la cuantificación del impacto de dichas noticias como su relevancia, su novedad, su impacto temporal y significancia. Todo esto es ofrecido con RavenPack Edge, que a través de una interfaz permite crear datasets, integrar los servicios mediante API o integrarlo con Snowflake. Los casos de uso típicos de esta herramienta son los de gestión de carteras, monitorización del mercado, gestión de riesgos, trading algorítmico, ...

En cuanto a la tecnología detrás del producto, RavenPack tiene una base de Procesamiento del Lenguaje Natural (NLP) capaz de procesar millones de documentos diarios y extraer de ellos datos estructurados. Para lograr esto, combina métodos de parsing lingüístico, desambiguación semántica y categorización de texto. Presenta modelos de machine learning propios para detectar sentimiento y otros atributos. Destaca el uso de autoencoders para perfeccionar su uso de los embeddings y el uso de NER [39] para el reconocimiento de entidades.

En definitiva, esta herramienta permite realizar un seguimiento en a diario del estado de las compañías a través de las noticias. Gracias a su tratamiento de las noticias de fuentes autorizadas, RavenPack News Analytics posibilita una estandarización de la información en línea con un enfoque sistemático fácilmente integrable. Sin embargo, no ofrece más que una primera capa de la funcionalidad de búsqueda puesto que no facilita la comprensión de los datos y falta una segunda capa que analice todos estos datos estandarizados.

LSEG – Machine Readable News

La división de análisis de datos del London Stock Exchange Group (LSEG) ofrece también un servicio de lectura de noticias. Machine Readable News [33] tiene en su núcleo un streaming de muy baja latencia de noticias, siendo de gran utilidad para el trading algorítmico. Con exclusividad de Reuters como fuente principal de datos, este servicio mantiene gran fiabilidad. También, se analizan distintas métricas como el sentimiento, la relevancia, la novedad y la significancia. No solo se centra en la velocidad del procesado, sino que ofrece un servicio llamado Headlines Direct que reduce la información, potenciando aún más la velocidad del sistema. A esto se le añaden más características que permiten la vigilancia del mercado en tiempo real.

En este caso también se utilizan algoritmos NLP y modelos de machine learning entrenados a lo largo de grandes conjuntos de datos históricos consistentes en noticias pasadas. LSEG Machine Readable News emplea un arsenal de técnicas NLP/ML especializadas en finanzas: desde los fundamentos (NER, clasificación, sentimiento) hasta métricas derivadas (novedad, impacto) y todo ello a velocidades altísimas. Su uso de IA está orientado a asegurar que cada noticia se entregue con el máximo de información útil ya calculada, lista para ser ingerida por algoritmos

o analistas sin procesamiento adicional.

Como se puede apreciar, la finalidad sigue siendo la toma de decisiones en tiempo real, priorizando la baja latencia y rapidez del servicio. No obstante, no aporta suficiente en el análisis fundamental de la empresa, pudiendo mejorar mucho si se incluyese un sistema que genere informes de estas noticias.

Alphasense

AlphaSense [1] es una plataforma de inteligencia de mercado y búsqueda impulsada por IA, enfocada en ayudar a profesionales financieros y corporativos a encontrar rápidamente información relevante entre millones de documentos. Presenta un contenido extenso con informes de analistas (esto incluye notas de análisis de acciones, informes sectoriales, recomendaciones, etc), transcripciones de conferencias, documentos de empresas, noticias, datos regulatorios y permite incluir contenido privado del cliente. Destaca su motor de búsqueda, que permite encontrar información fácilmente. Además, desde hace escasos meses, presentó las primeras soluciones con IA generativa que permiten mantener una interfaz de chat conversacional.

Las nuevas novedades publicadas por la empresa sugieren la implementación de un modelo de lenguaje propio, el ASLLM, desarrollado con su extenso catálogo de bases de datos. Se puede deducir el uso de RAG (Retrieval Augmented Generation) [30] en sus asistentes para la extracción de información. El buscador de AlphaSense no es un mero buscador de texto literal; utiliza procesamiento de lenguaje natural para entender la intención de la consulta y el contexto de los documentos. Implementa Smart Synonyms, lo que implica que se han entrenado modelos para encontrar términos relacionados semánticamente.

En conclusión, AlphaSense engloba muchas funcionalidades y empieza a incluir IA generativa en sus productos. Esto reduce drásticamente la fricción que se presenta en los anteriores productos, permitiendo al usuario despreocuparse de muchas etapas y pudiendo escribir peticiones con lenguaje natural directamente. A pesar de todos estos logros, no se ven avances con la arquitectura de agentes en la plataforma, algo que podría desbloquear el siguiente nivel para los analistas, cubriendo las fallas que los modelos de lenguajes presentan por sí solos.

FinChat.io

FinChat [33] ofrece una plataforma integral de análisis de empresas pensada para analistas de fondos de inversión: centraliza en una sola interfaz datos financieros (ingresos, márgenes, ratios, KPIs sectoriales) de más de 60 000 compañías globales, transcripciones de earnings calls y presentaciones corporativas, filings regulatorios (10-K, 10-Q, 8-K), estimaciones de analistas, datos de propiedad e incluso cartas

Funcionalidad	RavenPack	LSEG MRN	AlphaSense	FinChat.io
Análisis estructurado de noticias	☑	☑		
Búsqueda semántica de documentos			☑	
Chat conversacional / IA generativa			☑	☑
Resumen automático de documentos			☑	☑
Extracción de KPIs y métricas financieras			☑	☑
Generación de gráficos/tablas en respuestas				☑
Agentes IA autónomos	☐	☐	☐	☐

Figura 2.1: Comparativa de funcionalidades

de inversores destacados.

En cuanto a la tecnología aplicada, FinChat incorpora un modelo de conversación modificado con un Fine-Tuning al ámbito financiero (fundado en un LLM y técnicas de Retrieval-Augmented Generation), que fusiona un camino de ingesta e indexación de datos estructurados (números, series históricas, KPIs) con la generación de lenguaje natural para responder a preguntas en directo. Su Inteligencia Artificial, capacitada con informes, transcripciones y datos numéricos, no solo produce respuestas escritas, sino que también utiliza módulos especializados para cálculos y para la generación automática de tablas y diagramas.

Similar a AlphaSense, FinChat pretende incluir IA para incrementar la eficiencia y productividad de los analistas. A pesar de sus avances, no presenta agentes que asistan o hasta ayuden al analista haciendo tareas de forma autónoma, requiriendo siempre la presencia de un usuario, manteniendo una conversación con la IA para realizar las tareas paso a paso.

En conjunto, las cuatro soluciones analizadas (ver imagen 2.1) muestran cómo la IA aplicada al análisis financiero ha madurado desde la simple ingestión y estructuración de datos de noticias hasta interfaces conversacionales que facilitan la investigación. RavenPack News Analytics y LSEG Machine Readable News destacan por su capacidad de procesar información a muy baja latencia, ofreciendo métricas de sentimiento, relevancia, novedad e impacto que alimentan estrategias de trading algorítmico y monitorización en tiempo real. Sin embargo, ambas plataformas se detienen en la “primera capa” de datos estandarizados y carecen de mecanismos que transformen automáticamente esos flujos en informes o recomendaciones accionables.

Frente a ellas, AlphaSense y FinChat.io ilustran el salto hacia la generación de conocimiento en lenguaje natural. AlphaSense ha introducido su propio modelo (ASLLM) y técnicas de RAG para ofrecer búsquedas semánticas y respuestas conversacionales sobre un amplio repositorio de informes de analistas, transcripciones y documentos regulatorios. FinChat, por su parte, combina un LLM ajustado al

dominio financiero con módulos específicos para cálculos y creación automática de tablas y gráficos, centralizando datos cuantitativos y cualitativos en una única interfaz. Pese a sus avances, ambas plataformas aún no incorporan agentes autónomos capaces de ejecutar flujos de trabajo completos sin intervención constante.

Mirando al futuro, el valor diferencial radicará en integrar una segunda capa de síntesis avanzada (informes ejecutivos, alertas de riesgo o recomendaciones de trading) y en el despliegue de arquitecturas de agentes inteligentes que automaticen tareas complejas. Solo así estas soluciones podrán anticiparse a las necesidades de los analistas, ofrecer personalización a escala y cerrar el ciclo entre datos masivos y decisiones financieras efectivas. En definitiva, los analistas podrán desentenderse de procesos de obtención de datos y centrarse en el análisis más crítico y su posterior toma de decisiones basada en los datos obtenidos.

Capítulo 3

Fundamentos Teóricos

A lo largo de los siguientes capítulos se mencionarán diversas técnicas y procesos clave que requieren de cierta explicación para poder entender y contextualizar el trabajo realizado. En las siguientes secciones se analizarán técnicas como el Prompt Engineering y se desarrollarán procesos como el RAG, los agentes de inteligencia artificial y sus subprocesos más importantes. Todo ello, bajo el enfoque del procesamiento natural del lenguaje, en adelante NLP (Natural Language Processing). El NLP es el campo de la inteligencia artificial que se encarga de que las máquinas o modelos interpreten y trabajen con lenguaje humano.

3.1. LLM

Los Large Language Models o modelos grandes del lenguaje, en adelante LLM, han sido los protagonistas de la revolución en la IA generativa de los últimos años. Tras la publicación de los transformers en el paper de *Attention is all you need* [48], la investigación del sector ha explorado variaciones de dicha arquitectura, entrenando modelos cada vez más grandes hasta llegar a los LLM.

Primero, la arquitectura transformer es una red neuronal profunda o DNN (deep neural network) diseñada para procesar secuencias de datos, como en este caso, texto. A diferencia de modelos predecesores en el campo del NLP como las redes LSTM [24], los transformers presentan un mecanismo de atención muy potente. Este permite al modelo identificar y ponderar las partes más relevantes de, por ejemplo (simplificación), una secuencia de palabras en una oración para comprender el contexto global. Es decir, el modelo ve todas las palabras al mismo tiempo y es capaz de saber qué es lo más relevante, entendiendo el significado de la frase.

3.1.1. Embeddings

Es importante entender el concepto de los embeddings. Los modelos no entienden texto directamente sino que cada palabra se convierte en un vector numérico. Precisamente, no son palabras, sino que lo que convierte son *tokens*: una unidad básica de texto que un modelo de lenguaje procesa y puede ser una palabra, parte de una palabra o incluso un símbolo de puntuación. Entonces, los embeddings son estos vectores que capturan las relaciones semánticas de las palabras y son el punto de partida para la aplicación del mecanismo de atención. El espacio vectorial de estos embeddings tiene cientos o hasta miles de dimensiones para poder representar el significado de cada token. Por ejemplo, siguiendo con la simplificación de que un token puede ser una palabra, **príncipe** estaría cerca de **princesa** en este espacio vectorial, y además, el vector que une los vectores de estas dos palabras sería similar al que une **hombre** y **mujer**, puesto que las diferencias en significado son similares, esto de forma teórica, dependería del modelo de embeddings que se use.

3.1.2. Otras características de los LLM

Por otro lado, es necesario para el modelo tener una comprensión del orden de las palabras en cada oración puesto que afecta al significado global del texto. Por ello, los LLM usan codificación posicional o Positional Encoding, en adelante PE. Esta codificación le da noción al modelo de dónde están las palabras, es típico ver el PE con senos y cosenos (como en el paper original) aunque hay otras opciones.

La primera arquitectura propuesta de transformers se componía en dos principales partes: un encoder (encargado de procesar el texto de entrada) y un decoder (encargado de generar el texto de salida). A partir de ahí, han surgido variaciones con modelos como BERT [14] con solo encoder, o GPT-3 [7] con solo decoder.

Por último, los transformers permiten procesar todos los tokens al mismo tiempo, o lo que es lo mismo, paralelamente. Por ello, para el entrenamiento e inferencia de los LLM las GPUs (Graphics Processing Unit) se emplean ya que permiten la ejecución en paralelo como las operaciones de matrices, reduciendo significativamente la latencia de las respuestas y también porque estos LLM no caben en CPUs en muchas ocasiones.

3.1.3. Modelos via API

Puesto que ejecutar estos modelos requiere de un hardware muy costoso, para el proyecto se emplean APIs externas. Esto se traduce en que cada vez que en los flujos de trabajo se necesita de un LLM, se hará una petición a la Interfaz de Programación de Aplicaciones (API) de un proveedor de inteligencia de artificial.

El proveedor, como lo será en gran parte Openai [38] para este proyecto, aloja sus modelos y factura por el número de tokens usados por el usuario, con un precio que depende del modelo usado.

Esto permite usar tanto modelos propietarios, que no podrían ser accedidos de otra forma, como modelos abiertos pero desplegados en la infraestructura de una empresa externa. Por ende, una opción de alojar un modelo pequeño, aún posible en ordenadores con suficiente capacidad de computación y memoria, queda descartada puesto que su rendimiento no es suficiente. Primero, los modelos abiertos u open source, son todavía inferiores en cuanto a rendimiento. Segundo, más allá de que sea un modelo abierto o no, para poder alojar un modelo en un equipo local debe ser pequeño o estar cuantizado¹. Es decir, que en el caso de escoger desplegar un modelo para trabajar con él en local, sería como mucho un modelo de no más de 70 miles de millones de parámetros apróximadamente y cuantizado frente a elegir un modelo sin cuantizar del orden de los billones. Y, aunque más parámetros no signifique mayor rendimiento, sí que es un factor de peso. Por último, los costes iniciales para el despliegue son mucho mayores, al requerirse de tarjetas gráficas muy potentes y una memoria tan grande, todo esto dificulta alojar modelos en local. En conclusión, se decide usar modelos alojados en la nube y acceder a ellos por API para facilitar el desarrollo del proyecto.

3.2. RAG

Los LLM presentan varias deficiencias, entre ellas, es que están limitados en conocimiento en distintos aspectos. Cuando se publica un modelo nuevo los datos con los que se han entrenado el modelo solo llegan a ese momento de publicación. Por tanto, información posterior a ese día no estará en el conocimiento del modelo. Además, los datos empleados para el entrenamiento del LLM, aunque muy grandes, no son infinitos, por lo que hay mucha información con la que no ha sido entrenado. A esto se le añade que existen muchos documentos que no se encuentran al alcance de todos o son directamente privados. Toda esta ausencia de información se traduce en las populares alucinaciones, que ocurren cuando el modelo intenta generalizar en ocasiones que no conoce la respuesta y se inventa datos o respuestas.

Entonces, y aunque suene evidente, la mejor solución es incluir en el contexto que se envía al modelo la información necesaria para responder a la petición de manera correcta. Por ello, el Retrieval-Augmented Generation[30], en adelante RAG, pretende añadir esa información, que extrae de los documentos adjuntados. Este es un proceso de varias etapas:

¹La cuantización es una operación que consiste en reducir la precisión de los pesos de un modelo, por ejemplo, de 32 bits a 8 bits, para acelerar la inferencia y reducir el uso de memoria.

1. Creación de la base de datos: Proceso de creación de la base de datos para luego utilizar en inferencia. Este paso a de hacerse antes de la primera petición de usuario.
 - a) Base de conocimiento: lo primero es reunir los documentos necesarios para poder darle conocimiento al modelo. Es decir, en estos documentos, por grandes que sean, deben tener la información necesaria para responder a la pregunta de forma correcta.
 - b) Chunking: El texto se divide en pedazos. Normalmente, esto se realiza cada cierto número de palabras o caracteres. También es común ver solape en estos trozos de texto, se realiza para no perder el hilo y mantener un poco el contexto general. En la etapa del retrieval 4, se obtendrán estos pedazos, por lo que el tamaño de los mismos es importante. No debe ser ni muy pequeño para obtener información, ni muy grande para no obtener mucho ruido.
 - c) Vectorización : cada uno de estos pedazos de texto se pasa por un modelo de embeddings (explicado en la sección 3.1.1). Este modelo de embeddings devuelve un vector que representa el significado semántico del texto. En otras palabras, este vector muestra en números qué dice el texto. Estos vectores, se pueden comparar y hallar los más similares, que se traduce en que hablan de los mismo o aspectos muy cercanos
2. Petición del usuario: el usuario realiza una consulta al modelo.
3. Vectorización de la petición: esta consulta se pasa por el modelo de embeddings y se convierte en vector. Es importante recalcar que el modelo empleado para la vectorización de la consulta debe ser el mismo que el que haya sido usado para vectorizar la base de conocimiento. Si no es así, las siguientes etapas no tendrían sentido.
4. Búsqueda por similitud (Retrieval): Con el vector que representa la consulta, se compara con los vectores de todos los pedazos de texto de los documentos de la base de datos. Normalmente, se recogen de 3 a 5 trozos, los más similares al vector de la petición. Esta similitud se puede determinar de distintas formas; la más común es por similitud del coseno. Consiste en realizar entre dos vectores y luego dividir entre la multiplicación de sus módulos como vemos en la siguiente figura:

$$\text{cosine similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

5. Generación de contexto: tras obtener los pedazos más relevantes, se extrae en su forma de texto y se le pasan al modelo como contexto. Es importante indicarle al modelo que estos pedazos de texto son en lo que tiene que basar sus respuestas.
6. Respuesta del modelo: el modelo responderá a la petición del usuario, fundamentando su generación en los trozos que se le hayan pasado en el contexto.

Este proceso es el RAG más popular y sencillo. A lo largo del trabajo realizado se explora cómo optimizar cada etapa y cómo adaptar cada paso para obtener mejores respuestas, más consistentemente y con más documentos a la vez. Como se ha demostrado, el RAG es fundamental para reducir alucinaciones y con un buen proceso, la inteligencia artificial aumenta su utilidad radicalmente, reduciendo sus vulnerabilidades drásticamente.

3.3. Prompt Engineering

El prompt es el nombre anglosajón para la consulta que se le envía al modelo, entonces en español se le podría atribuir el nombre de ingeniería de consultas. Esta categoría surge tras la aparición de los LLM, especialmente por ser fundamentalmente modelos probabilísticos. En consecuencia, la ingeniería de prompts es una disciplina que pretende aumentar la probabilidad de obtener mejores respuestas, más precisas, más alineadas con la estructura de salida que se busca, y con menos alucinaciones a partir de diseñar cuidadosamente los prompts. En este proyecto se emplearán múltiples técnicas de prompting, a continuación se detallan las principales. Antes de comenzar con la explicación de las técnicas es importante destacar que el modelo no se actualiza durante este proceso, utiliza únicamente el contexto proporcionado (inferencia).

En primer lugar, hay que comprender que los LLM tienen una ventana de contexto de entrada limitada y hay que optimizar la información que se le pasa de entrada. Aunque los nuevos modelos presentan cada vez más ventanas de entrada de órdenes superiores, estudios demuestran la caída de precisión a medida que se aumenta la cantidad de tokens de entrada. En concreto, el test NIAH² fue realizado a 12 LLM, mostrando cómo con un tamaño de 32 mil tokens, el modelo (sin ayuda de un sistema RAG) no era capaz de responder adecuadamente más de un 50 %

²El test NIAH (“needle-in-a-haystack”) consiste en encubrir una pieza muy pequeña de información (la “aguja”) dentro de un contexto mucho más extenso (“el pajar”) y exigir al modelo que la localice respondiendo a una pregunta. En la práctica, se inserta un enunciado clave en pasajes largos de texto y se mide hasta qué punto el modelo puede recuperarlo, sin apoyarse en solapamientos léxicos directos entre pregunta y contexto

de las veces [34]. Por tanto, se debe reducir el tamaño del prompt manteniendo lo imprescindible por razones de coste y de rendimiento.

Sabiendo que el prompt debe ser conciso, todo lo que se incluya debe de estar por una razón. Por ende, se debe ser directo y evitar expresiones de cortesía que no solo añaden tokens de entrada sino que también puede aumentar los de salida, al hacer que el LLM intente replicar tu estilo cortés. Además, hay que usar verbos imperativos, eligiendo verbos afirmativos y dejando muy claro la tarea del modelo [8].

Por otro lado, en este proyecto especialmente, se inyectarán muchas indicaciones, contextos y variables en el prompt de forma automática. Por lo que es de vital importancia usar delimitadores para marcar correctamente el inicio y fin de los campos en el prompt. Por ejemplo, incluir delimitadores antes y después de la pregunta del usuario hará que el LLM sepa cuál es exactamente la pregunta y no lo confunda con otras partes de las instrucciones. Esto es a su vez útil para evitar "prompt injection"[42], ataque muy similar al de Sql injection³, que permite al usuario dar instrucciones al LLM como administrador y atacar los procesos internos de la aplicación IA. Al estar estos delimitadores el LLM puede identificar estos intentos de sabotage viendo como no son instrucciones del sistema del usuario.

Otra técnica que mejora considerablemente el rendimiento de los modelos y afina mejor el formato de salida de su respuesta es el One-Shot o Few-Shot prompting [7]. Con shot se refiere a ejemplos, es decir, one-shot es incluir un ejemplo, few-shot más de uno y zero-shot significa no incluir ninguno. Esto tiene sentido puesto que el modelo recibe más contexto de la petición del usuario y ve una o más posibles repuestas a peticiones similares. También, especificar el rol del LLM hace que se comporte como tal, ajusta su tono y devuelve respuestas más acertadas a ese ámbito. Por ejemplo, en el caso de este proyecto, es útil especificarle que es un analista financiero. Con esto, el modelo centra su mensaje en métricas y resultados financieros en vez de describir a las empresas desde otro punto de vista, otorgando mayor profundidad en lo que de verdad importa en el análisis de empresas.

3.3.1. Modelos de razonamiento

-CoT [50] +Self-Consistency +Tree of Thought = modelos de razonamiento

Es importante recalcar que la disciplina del prompt engineering está en constante evolución. Puesto que se pretende mejorar el rendimiento de los modelos y estos están en constante cambio y mejora, el prompt engineering va cambiando constantemente. Previamente, se mencionó cómo el Chain of Thought (Cot) [50]

³La inyección SQL es una técnica de ataque en la que un atacante inserta (inyecta) código SQL malicioso en las entradas de una aplicación (formularios, URLs, cabeceras) para que la base de datos lo ejecute. Con ello puede leer, modificar o eliminar datos de forma no autorizada, e incluso tomar control del sistema.

llevó a los modelos de razonamiento, pues estos modelos, por ejemplo, requieren de un prompting muy distinto.

Antes de nada, el CoT consiste en que pedirle al LLM que en la respuesta haga paso a paso la tarea que se le pide. Esto se puede hacer tanto incluyendo en el prompt un ejemplo de como hacer esto (One-shot CoT) como directamente pedirle que piense paso a paso (Zero-shot CoT). El estudio que publicó CoT como una estrategia que mejoraba las respuestas de los LLM por parte del departamento de investigación de Google demuestra cuantitativamente su mejora sobre el prompting usual. Con un benchmark matemático GSM8K, el modelo PALM de 540 mil millones de parámetros pasaba de tener un 18 % de aciertos a un 57 % al incluir esta técnica. El CoT es una de las características principales de los modelos de razonamiento, estos modelos son entrenados con ejemplos de cadenas de pensamiento para que no solo luego el LLM sea capaz de hacer CoT automáticamente sino que también lo haga de forma óptima.

Además, los nuevos modelos de razonamiento no solo lanzan una CoT sino que algunos incorporan varias cadenas y luego eligen la mejor. Hay diversos enfoques a esta funcionalidad, como el Tree-ofThought (ToT) [52]. El método ToT expande el proceso de razonamiento de un modelo de lenguaje explorando múltiples “ramas” de pensamiento en forma de árbol, en lugar de generar una única secuencia lineal de tokens. Cada nodo del árbol representa un estado intermedio (un “pensamiento”) y se evalúa según criterios de calidad o relevancia, permitiendo al sistema retroceder y explorar alternativas hasta encontrar la cadena de razonamientos más prometedora para resolver tareas complejas.

Entonces, estos modelos, al incorporar ya varias técnicas de generación de tokens de pensamiento, no necesitan tanto prompt. Es diferente, el modelo razona y consigue entender el contexto más fácilmente. Aunque esto no significa que las otras técnicas no ayuden.

Capítulo 4

Funcionalidad de Búsqueda Local

Los modelos de lenguaje o LLM (Large Language Models) no son perfectos y, en numerosas ocasiones, generan respuestas erróneas, fenómeno conocido como *alucinaciones*. Estas alucinaciones surgen porque la generación de respuestas de los LLM se basa en patrones estadísticos aprendidos durante su entrenamiento. Por lo tanto, cuando la información necesaria para responder a una consulta no estuvo presente en el conjunto de datos de entrenamiento, el modelo tiende a generalizar y produce una respuesta que, aunque puede parecer plausible, es incorrecta.

En el contexto del análisis de empresas, es frecuente que mucha información relevante haya sido publicada después del entrenamiento del modelo, no haya sido incluida en el conjunto de datos o, simplemente, no sea de dominio público. Por esta razón, es fundamental proporcionar al modelo de inteligencia artificial información actualizada y de calidad para garantizar la generación de respuestas precisas y confiables.

4.1. RAG con metadatos enriquecidos

Como primera solución, se propone un flujo de trabajo basado en RAG (Retrieval-Augmented Generation) [30]. Este pipeline permite que el modelo reciba información relevante relacionada con la consulta del usuario, lo que facilita la generación de respuestas no solo más precisas, sino también fundamentadas. A diferencia de un enfoque manual, en el que el analista debe buscar información específica en los documentos de la empresa, este flujo de trabajo optimiza el acceso a los datos, proporcionando un método más rápido y eficiente para analizar la información empresarial.

Además, para optimizar aún más la búsqueda, los documentos serán filtrados según sus metadatos. Esto implica que los documentos se categorizarán según diversos parámetros, de manera que, al realizar una consulta sobre una categoría

específica, la búsqueda se limite únicamente a los documentos correspondientes a dicha categoría. Este enfoque no solo facilita la localización de información relevante, sino que también reduce el gasto computacional al evitar búsquedas innecesarias en el conjunto completo de documentos. El metadato más evidente tal vez sea el del año. Puesto que muchos documentos son anuales y son publicados año tras año, catalogarlo para su año es relativamente accesible de hacer y reduce la búsqueda rápidamente.

4.1.1. Objetivos de la Prueba de Concepto

El propósito de esta prueba de concepto es el de completar los objetivos enunciados a continuación. Como principales objetivos de este flujo de trabajo se encuentran los siguientes:

- Dotar al LLM de un contexto completo para el análisis empresarial.
- Reducir las alucinaciones del modelo.
- Establecer una alternativa más rápida a la búsqueda manual de información de una empresa.
- Comprobar la veracidad de las respuestas.

Estos puntos serán evaluados de la mano de expertos analistas con años de experiencia en el sector para demostrar que esta herramienta es útil para esta aplicación y para comprobar la mejora frente a un modelo de inteligencia artificial por sí solo.

4.1.2. Descripción del Workflow

En esta prueba de concepto se utiliza como ejemplo la empresa Logista para poder hacer consultas sobre la misma.

Este flujo de trabajo presenta tres partes, descritas en la figura 4.1. La primera, correspondiente a la rama izquierda de la figura, es la ingesta de datos. A partir de casi 400 páginas de documentos PDF de la empresa Logista se genera una base de datos vectorizada. En concreto, estos documentos son inicialmente divididos en fragmentos de texto o chunks. Estos trozos de texto son posteriormente procesados por un modelo de embeddings que genera vectores para representar el significado semántico de cada chunk. Entonces, cuando la petición del usuario sea recibida y a su vez convertida a vector por el mismo modelo de embeddings, se hará una búsqueda por similitud y se obtendrán los k (en este caso, $k=10$) fragmentos más relevantes, que se añadirán al prompt como contexto para el modelo final.

La segunda etapa consiste en la categorización o tagging de la consulta. En esta fase, la petición del usuario es procesada por un primer modelo que clasifica la

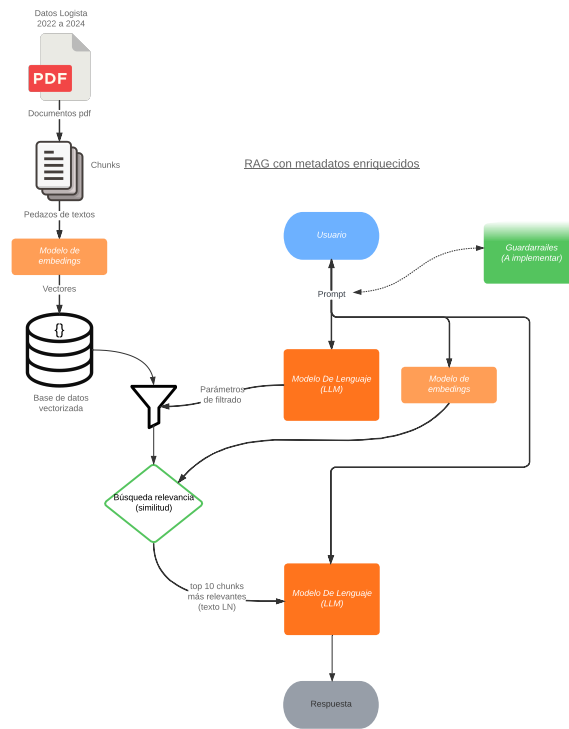


Figura 4.1: Diagrama de flujo de RAG con metadatos enriquecidos.

solicitud y genera parámetros estructurados para el filtrado de datos. Esto permite aplicar un filtro previo a la búsqueda por similitud mencionada en la etapa anterior, enfocando la búsqueda únicamente en los documentos más relevantes para la consulta.

Finalmente, en la tercera etapa, se realiza una segunda llamada al modelo de lenguaje. Este recibe la consulta del analista junto con el contexto obtenido de la base de conocimiento vectorizada.

4.1.3. Especificaciones técnicas

Ya que el objetivo principal de esta fase es evaluar el sistema, no es una prioridad probar diferentes modelos de lenguaje a lo largo de los distintos tipos de RAG. En este caso, los LLM necesarios se implementan mediante llamadas a la API de OpenAI. Específicamente, se emplea el modelo *gpt-4o-mini* para el desarrollo y *gpt-4o* para las pruebas finales. Se mantiene el valor del parámetro temperatura igual a 0 para que el modelo sea lo más determinista posible se busca evitar que el

modelo sea muy creativo para evitar alucinaciones. Esta elección ofrece múltiples ventajas frente a otras formas de implementación o modelos. Principalmente, para las primeras pruebas de concepto, no tener que desplegar un modelo garantiza una gran facilidad y rapidez a la hora de hacer pruebas, evitando preocupaciones sobre las necesidades computacionales del LLM. Además, los modelos propietarios de OpenAI ofrecen buenos resultados, aunque este análisis se realizará más adelante.

Por otra parte, en la etapa de ingesta de datos, el chunking o división del texto se realiza cada 1.000 caracteres y sin solape. Para la vectorización, el modelo de embeddings *text-embedding-ada-002* se encarga de convertir los fragmentos de texto en vectores. Este modelo proporciona una representación de 1.536 dimensiones y es importante recalcar que el modelo de embeddings debe ser el mismo que el empleado para vectorizar la petición del usuario para que el retrieval funcione adecuadamente. A continuación, se necesita una base vectorial para almacenar estos vectores y garantizar su persistencia para futuras pruebas. En este caso, ChromaDB [13] fue el software empleado para este proceso.

Por último, en la etapa del retrieval u obtención de la información relevante de esta prueba, se utiliza una búsqueda por similitud para obtener los $k=10$ chunks más similares a la petición del usuario. Como se mencionó anteriormente, en esta etapa la petición se vectoriza con el mismo modelo de embeddings y luego se compara con los vectores de la base de datos para extraer los más relevantes.

4.1.4. Resultados

En la primeras pruebas del modelo, observamos que las respuestas son bastante completas y proporcionan detalle acerca de las preguntas acerca de la empresa estudiada. Por el momento, se valoran los resultados analizando las respuestas obtenidas. Uno de los pilares de una empresa es su gente, por ello, en el análisis de empresas, el estudio de la directiva y, en especial, el consejero Delegado y sus incentivos es muy importante. Una de las primeras pruebas ha sido la de analizar el CEO de Logista, como se puede observar en la figura 4.2 el resultado es muy positivo. Se obtiene información del actual Consejero Delegado, mencionando fechas relevantes, describiendo su carrera profesional previa a trabajar en Logista y su formación académica.

Sin embargo, observamos con facilidad la ausencia de información de datos incluidos en tablas. Esto es crítico puesto que en el análisis de las empresas se deben revisar muchas tablas de contabilidad como el balance de situación, flujo de caja, etc.

Respuesta:

El actual Consejero Delegado de Logista Integral, S.A. es D. Íñigo Meirás Amusco, quien ocupa este cargo desde diciembre de 2019 y fue reelegido por la Junta General de Accionistas el 2 de febrero de 2024. Su trayectoria profesional es notable y se caracteriza por una amplia experiencia en el sector de la construcción y los servicios, así como en la gestión de grandes empresas.

Trayectoria Profesional

D. Íñigo Meirás comenzó su carrera en Ferrovial en 1992, donde desempeñó diversos roles que le permitieron adquirir una sólida experiencia en la gestión empresarial. A lo largo de su carrera en Ferrovial, ocupó posiciones clave, incluyendo la de Consejero Delegado de Ferrovial Servicios y Consejero Delegado de Ferrovial Aeropuertos. En 2009, fue nombrado CEO del grupo Ferrovial, cargo que mantuvo hasta septiembre de 2019. Esta experiencia en una de las principales empresas de infraestructuras y servicios de España le ha proporcionado una visión integral del negocio y una capacidad de liderazgo destacada.

Antes de su paso por Ferrovial, D. Íñigo Meirás también trabajó en el Grupo Carrefour y en Holcim, lo que le permitió diversificar su experiencia en diferentes sectores, desde la distribución hasta la construcción.

Formación Académica

D. Íñigo Meirás es Licenciado en Derecho por la Universidad Complutense de Madrid y cuenta con un MBA por el Instituto de Empresa (IE). Esta formación académica le ha dotado de una sólida base en aspectos legales y de gestión empresarial, lo que es fundamental para su rol como Consejero Delegado en una empresa que opera en un entorno regulado y competitivo.

Figura 4.2: Respuesta: Análisis Consejero Delegado.

4.1.5. Aspectos a mejorar

Puesto que en el análisis de las empresas se trabaja con muchas tablas y cada una con su formato, es habitual tener múltiples en los documentos usados. Además, son difíciles de procesar debido a su gran variabilidad. De cara a siguientes iteraciones, el poder leerlas por código correctamente se establece como una prioridad tras el feedback de los analistas. Esto es porque muchas peticiones útiles que ellos harían serían acerca de evoluciones de resultados, números específicos en varios documentos, etc. Por lo que no es un simple detalle para la herramienta en desarrollo.

4.1.6. Conclusión

La implementación del pipeline RAG con metadatos enriquecidos ha demostrado ser una solución eficaz para abordar uno de los principales retos de los modelos de lenguaje: la generación de respuestas alucinadas o incorrectas. Al proporcionar al modelo contexto específico y actualizado, derivado directamente de documentos empresariales relevantes, se mejora notablemente la precisión, trazabilidad y utilidad de las respuestas generadas.

Esta prueba de concepto, aplicada al análisis de la empresa Logista, ha permitido validar que el sistema puede recuperar y estructurar información crítica (como la trayectoria del CEO) de manera coherente y detallada. Además, se ha demostrado que esta herramienta proporciona una alternativa más eficiente al proceso manual de búsqueda documental, acelerando el análisis sin comprometer la calidad del mismo.

Uno de los principales avances técnicos ha sido la incorporación del filtrado por metadatos, especialmente por año, lo que ha permitido afinar la búsqueda y reducir la carga computacional. También se ha podido comprobar que, a pesar de sus fortalezas, el sistema actual aún presenta limitaciones en el tratamiento de información tabular, un aspecto clave para el análisis financiero profesional. Este será un foco de mejora en las siguientes iteraciones, especialmente ante la necesidad de interpretar tablas contables y series temporales con mayor fiabilidad.

De cara al futuro, y en línea con los estándares del sector, se adoptará el inglés como idioma por defecto para los siguientes desarrollos, lo cual facilitará su escalabilidad, compatibilidad y posible integración con herramientas externas. Asimismo, cabe destacar que esta arquitectura permite trabajar de forma segura con documentación privada de gran valor, integrando información confidencial en los procesos de análisis sin exponerla a modelos generalistas o públicos.

En definitiva, este flujo RAG no solo incrementa la eficiencia del proceso analítico, sino que representa un paso sólido hacia la creación de agentes inteligentes capaces de asistir a analistas financieros con información precisa, actual y perso-

	Share Capital	Share Premium	Reserves of the Parent	Reorganisation Reserves	Reserves at Consolidated Companies	Exchange Rate Changes	Reserve for First-Time Application of IFRSs	Consolidated Profit for the Year	Interim Dividend	Treasury Shares	Equity Attributable to Shareholders of the Parent	Minority Interests	Total Equity
BALANCE AS OF SEPTEMBER 30, 2021	26,550	867,808	289,014	(753,349)	(31,735)	(97)	19,950	173,961	(54,116)	(14,346)	523,640	834	524,474
Net profit attributable to the Parent	-	-	-	-	-	-	-	198,848	-	-	198,848	-	198,848
Exchange Rates Changes	-	-	-	-	-	(354)	-	-	-	-	(354)	-	(354)
Loss attributable to minority interests	-	-	-	-	-	-	-	-	-	-	-	299	299
Actual gains/losses	-	-	-	-	6,778	-	-	-	-	-	6,778	-	6,778
Income and expenses recognised in the period	-	-	-	-	6,778	(354)	-	198,848	-	-	205,272	299	205,571
Transactions with Shareholders:	-	-	-	-	-	-	-	-	-	-	-	-	-
Distribution of profit:	-	-	-	-	-	-	-	-	-	-	-	-	-
To reserves	-	-	113,073	-	(102,684)	-	-	(10,389)	-	-	-	-	-
To dividends (Note 14.e)	-	-	-	-	-	-	-	(163,572)	54,116	-	(109,456)	-	(109,456)
Dividends (Note 14.e)	-	-	-	-	-	-	-	-	(56,714)	-	(56,714)	-	(56,714)
On treasury shares operations (Note 14.b and 14.f)	-	-	718	-	-	-	-	-	-	(3,917)	(3,199)	-	(3,199)
Incentive Plan (Note 4.12)	-	-	768	-	-	-	-	-	-	1,663	2,431	-	2,431
Business combinations	-	-	-	-	-	-	-	-	-	-	-	3,691	3,691
Others	-	-	-	-	2	-	-	-	-	-	2	(105)	(103)
BALANCE AS OF SEPTEMBER 30, 2022	26,550	867,808	403,573	(753,349)	(127,639)	(451)	19,950	198,848	(56,714)	(16,600)	561,976	4,719	566,695
Net profit attributable to the Parent	-	-	-	-	-	-	-	272,254	-	-	272,254	-	272,254
Exchange Rates Changes	-	-	-	-	-	107	-	-	-	-	107	-	107
Loss attributable to minority interests	-	-	-	-	-	-	-	-	-	-	-	410	410
Actual gains/losses	-	-	-	-	2,530	-	-	-	-	-	2,530	-	2,530
Income and expenses recognised in the period	-	-	-	-	2,530	107	-	272,254	-	-	274,891	410	275,301
Transactions with Shareholders:	-	-	-	-	-	-	-	-	-	-	-	-	-
Distribution of profit:	-	-	-	-	-	-	-	-	-	-	-	-	-
To reserves	-	-	6,794	-	10,084	-	-	(16,878)	-	-	-	-	-
To dividends (Note 14.e)	-	-	-	-	-	-	-	(181,970)	56,714	-	(125,256)	(36)	(125,292)
Dividends (Note 14.e)	-	-	-	-	-	-	-	-	(64,619)	-	(64,619)	-	(64,619)
On treasury shares operations (Note 14.b and 14.f)	-	-	4,804	-	-	-	-	-	-	(7,332)	(2,528)	-	(2,528)
Incentive Plan (Note 4.12)	-	-	(118)	-	-	-	-	-	-	2,549	2,549	-	2,549
Business combinations (Note 7.2)	-	-	(56,223)	-	-	-	-	-	-	-	(56,223)	(489)	(56,712)
Others	-	-	-	-	(83)	-	-	-	-	-	(83)	-	(83)
BALANCE AS OF SEPTEMBER 30, 2023	26,550	867,808	358,830	(753,349)	(115,108)	(344)	19,950	272,254	(64,619)	(21,265)	590,707	4,604	595,311

Figura 4.3: Tabla original usada para la comparativa.

nalizada.

4.1.7. Problemática en el procesado de documentos

Tras las primeras interacciones de desarrollo, debido al uso mayoritario de documentos PDF, los resultados no son los esperados. Los documentos PDF añaden una dificultad extra a la hora de procesarlos en el pipeline de RAG. Al convertir los documentos a texto plano para la posterior separación en chunks y su debida vectorización, se observa cómo se pierde la estructura de las tablas, imágenes y demás elementos visuales. En concreto, es crítico para un análisis empresarial detallado poder tener la información incluida en tablas contables, puesto que es donde se representan los datos específicos del balance de la compañía, resultados, flujo de caja, etc.

4.1.8. Implementación actual

Para comparar el desempeño de las distintas librerías y herramientas externas, usaremos la tabla de la figura 4.3 observando cómo es procesada por las distintas herramientas. Este análisis fue realizado en detalle con muchas más tablas, se puede ver todo la comparativa en el notebook subido, especificado en el anexo B.

Mediante el uso de la librería PyPDF2 [20] se extrae el texto de los documentos PDF. Como se puede observar en la figura 4.4, a pesar de extraer correctamente el texto, las tablas pierden su estructura y se obtiene la misma de una forma ilegible para los modelos. De este modo, no es posible obtener dicha información de las tablas mediante el RAG puesto que no se puede determinar qué valor pertenece a qué fila o columna. Si se presta suficiente atención, aun con una etapa posterior para limpiar la tabla, las columnas están descuadradas y varios valores aparecen

		Reserve for Equity																								
		Reserves	Reserves at	First-Time	Consolidated	Attributable																				
Share	Share of the	Reorganisation	Consolidated	Exchange Rate	Application	Profit	Interim	Treasury	to Shareholders	Minority	Total															
Capital	Premium	Parent	Reserves	Companies	Changes	of IFRSs	for the Year	Dividend	Shares	of the Parent	Interests	Equity														
BALANCE AS OF	SEPTEMBER 30,	2021	26,550	867,808	289,014	(753,349)	(31,735)	(97)	19,950	173,961	(54,116)	(14,346)	523,640	834	524,474											
Net profit attributable to the Parent		-	-	-	-	198,848	-	198,848	-	198,848																
Exchange Rates Changes		-	-	-	(354)	-	(354)	-	(354)																	
Loss attributable to minority interests		-	-	-	-	-	-	-	-	299	299															
Actuarial gains/ losses		-	-	-	6,778	-	6,778	-	6,778																	
Income and expenses recognised in the period		-	-	-	6,778	(354)	-	198,848	-	205,272	299	205,571														

Figura 4.4: Tabla procesada con PyPDF2 [20]

	Share\nCapital	Share\nPremium	Reserves\nof the\nParent	None	None	Reorganisation\nReserves	None	None	Reserves at\nConsolidated\nCompanies	...
BALANCE AS OF SEPTEMBER 30, 2021\nNet profit a...			None	None	None	None		None	None	None
1	None	26,550	867,808	289,014	None	None	(753,349)	None	None	(31,735) ...
2	None	-\\n-\\n-\\n-	-\\n-\\n-\\n-	-\\n-\\n-\\n-	None	None	-\\n-\\n-\\n-	None	None	-\\n-\\n- ...
3	None	None	None	None	None	None	None	None	None	...
4	None	-	-	-	None	None	-	None	None	6,778 ...
5	None	-\\n-\\n-\\n-\\n-\\n-\\n-	-\\n-\\n-\\n-\\n-\\n-\\n-	113,073\\n-	None	None	-\\n-	None	None	...

Figura 4.5: Tabla procesada con pdfplumber [45]

en otras casillas.

4.1.9. Soluciones implementadas

De cara a solventar esta problemática, se implementaron distintas soluciones existentes que pretendían obtener las tablas con OCR usando los bordes y separadores de las mismas para extraer los valores con su debida correspondencia. A continuación, se aprecian los resultados obtenidos usando pdfplumber [45], tabula [3], camelot [9] en la figura 4.5, 4.7, y 4.6 respectivamente. Las tablas ya en formato dataframe, preservan cierta estructura pero, los resultados no son suficientemente buenos. Sobre todo, los encabezados o títulos de filas y columnas se pierden, esto, junto con errores de valores mal asignados no permite el uso de estas tablas en el pipeline de RAG correctamente. Estos resultados empeoran cuando en el formato original de las tablas en los PDFs, no existen separadores por cada fila, lo que dificulta que el OCR funcione adecuadamente. La gran variabilidad de una empresa a otra en cuanto a presentación de resultados en formas tabulares lleva a que estas soluciones no sean suficientes.

0								Reserve					
1								for	Equity				
2	Reserves				Reserves at			First-Time	Consolidated			Attributable	
3	Share	Share	of the	Reorganisation	Consolidated	Exchange Rate Application			Profit	Interim	Treasury	to Shareholders	Minority
4	Capital	Premium	Parent	Reserves	Companies	Changes	of IFRSs		for the Year	Dividend	Shares	of the Parent	Interests
5	BALANCE AS OF SEPTEMBER 30, 2021	26,550	867,808	289,014	(753,349)	(31,735)	(97)	19,950	173,961	(54,116)	(14,346)	523,640	834
6	Net profit attributable to the Parent	-	-	-	-	-	-	-	198,848	-	-	198,848	-
7	Exchange Rates Changes	-	-	-	-	-	(354)	-	-	-	-	(354)	-
8	Loss attributable to minority interests	-	-	-	-	-	-	-	-	-	-	-	299
9	Actuarial gains/losses	-	-	-	-	6,778	-	-	-	-	-	6,778	-

Figura 4.6: Tabla procesada con camelot [9]

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Reserve	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	for	NaN	NaN	NaN	Equit
1	NaN	NaN	NaN	Reserves	NaN	Reserves at	NaN	First-Time	Consolidated	NaN	NaN	Attributabl
2	NaN	Share	Share	of the	Reorganisation	Consolidated	Exchange Rate	Application	Profit	Interim	Treasury	Shareholder
3	NaN	Capital	Premium	Parent	Reserves	Companies	Changes	of IFRSs	for the Year	Dividend	Shares	of the Parent
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	BALANCE AS OF SEPTEMBER 30, 2021	26,550	867,808	289,014	(753,349)	(31,735)	(97)	19,950	173,961	(54,116)	(14,346)	523,640
6	Net profit attributable to the Parent	-	-	-	-	-	-	-	198,848	-	-	198,848
7	Exchange Rates Changes	-	-	-	-	-	(354)	-	-	-	-	(354)
8	Loss attributable to minority interests	-	-	-	-	-	-	-	-	-	-	-
9	Actuarial gains/losses	-	-	-	-	6,778	-	-	-	NaN	NaN	6,778

Figura 4.7: Tabla procesada con tabula [3]

4.1.10. Solución final

Tras evaluar las posibles soluciones mediante el uso de OCR a través de librerías de python, se observa la necesidad de ir más allá para solucionar este problema. Otra tecnología aún más reciente es la de los modelos de visión o LVM (Large Vision Model)¹.

Para facilitar la implementación de estos modelos, se usa Llama Parse [32], servicio de Llama Index [31]. El acceso al servicio es mediante API y ofrece cierta parametrización junto con distintos modos de procesado. A continuación, se puede observar el rendimiento tanto en el modo estándar 4.8 como en el avanzado 4.9. El modo estándar ya mejora notoriamente a las librerías usadas previamente, tan solo se destacan pequeños errores que lo alejan de la perfección. En este ejemplo (figura 4.8), se ve cómo todo está bien ordenado, pero "Years ended" que iba encima de la tabla, está como una columna nueva. Esto no cambia prácticamente el significado de la tabla ni altera ningún valor, quedándose muy cerca de la perfección.

Por otro lado, el modo avanzado calca las tablas, ya sean más o menos grandes. Sobre todo, se nota mejora con tablas sin muchos delimitadores, donde los LVM son mucho mejores que un simple sistema OCR.

Es importante destacar que esta herramienta permite obtener los documentos procesados en distintos formatos. Elegimos el *markdown*² para poder apreciar la jerarquía del mensaje visualmente.

Además, para afinar el procesado con Llama Parse al máximo y usar este software de la manera más eficiente posible, se emplea un modo automático. De esta forma, el documento a procesar se analiza rápidamente el texto y cuando se detecta una tabla, entonces se combinan LVM con LLM y OCR. Entonces, el sistema no pierde tiempo ni dedica recursos innecesarios a los párrafos de texto y reserva los recursos más demandantes a nivel computacional para las tablas.

En conclusión, este problema al procesar las tablas en PDFs es resuelto con Llama Parse. Esta solución añade una etapa en la ingesta de datos, pero asegura la comprensión de las tablas de contabilidad de forma correcta. Cuando se reciban

¹Los modelos de visión orientados a la lectura de tablas son arquitecturas especializadas dentro del campo del aprendizaje profundo diseñadas para interpretar, estructurar y extraer datos tabulares directamente desde imágenes o documentos en formato visual, como archivos PDF escaneados. A diferencia de los enfoques tradicionales basados en OCR (Reconocimiento Óptico de Caracteres), que suelen limitarse a extraer texto sin contexto estructural, estos modelos combinan capacidades de visión por computadora y procesamiento del lenguaje natural para identificar celdas, encabezados, relaciones fila-columna y estructuras jerárquicas dentro de una tabla. Algunos modelos recientes, conocidos como Large Vision Models (LVMs), integran mecanismos multimodales que permiten razonar sobre los elementos visuales (líneas, bordes, posición) y semánticos (tipo de datos, unidades, encabezados) al mismo tiempo.

²Markdown es un lenguaje de marcado ligero diseñado para escribir texto de forma sencilla y legible que pueda convertirse fácilmente en HTML u otros formatos.

	Years ended		
	September 30, 2023	September 24, 2022	September 25, 2021
Net income	\$ 96,995	\$ 99,803	\$ 94,680
Other comprehensive income/(loss):			
Change in foreign currency translation, net of tax	(765)	(1,511)	501
Change in unrealized gains/losses on derivative instruments, net of tax:			
Change in fair value of derivative instruments	323	3,212	32
Adjustment for net (gains)/losses realized and included in net income	(1,717)	(1,074)	1,003
Total change in unrealized gains/losses on derivative instruments	(1,394)	2,138	1,035
Change in unrealized gains/losses on marketable debt securities, net of tax:			
Change in fair value of marketable debt securities	1,563	(12,104)	(694)
Adjustment for net (gains)/losses realized and included in net income	253	205	(273)
Total change in unrealized gains/losses on marketable debt securities	1,816	(11,899)	(967)
Total other comprehensive income/(loss)	(343)	(11,272)	569
Total comprehensive income	\$ 96,652	\$ 88,531	\$ 95,249

	Years ended	September 30, 2023	September 24, 2022	September 25, 2021
Net income		\$ 96,995	\$ 99,803	\$ 94,680
Other comprehensive income/(loss):				
Change in foreign currency translation, net of tax		(765)	(1,511)	501
Change in unrealized gains/losses on derivative instruments, net of tax:				
Change in fair value of derivative instruments		323	3,212	32
Adjustment for net (gains)/losses realized and included in net income		(1,717)	(1,074)	1,003
Total change in unrealized gains/losses on derivative instruments		(1,394)	2,138	1,035
Change in unrealized gains/losses on marketable debt securities, net of tax:				
Change in fair value of marketable debt securities		1,563	(12,104)	(694)
Adjustment for net (gains)/losses realized and included in net income		253	205	(273)
Total change in unrealized gains/losses on marketable debt securities		1,816	(11,899)	(967)
Total other comprehensive income/(loss)		(343)	(11,272)	569
Total comprehensive income		\$ 96,652	\$ 88,531	\$ 95,249

Figura 4.8: Comparativa de otra tabla procesada con LlamaParse [32] en modo estándar (formato *markdown*)

	Share Capital	Share Premium	Reserves of the Parent	Reorganisation Reserves	Reserves at Consolidated Companies	Exchange Rate Changes	Reserve for First-Time Application of IFRSs	Consolidated Profit for the Year	Interim Dividend	Treasury Shares	Equity Attributable to Shareholders of the Parent	Minority Interests	Total Equity
BALANCE AS OF SEPTEMBER 30, 2021	26,550	867,808	289,014	(753,349)	(31,735)	(97)	19,950	173,961	(54,116)	(14,346)	523,640	834	524,474
Net profit attributable to the Parent	-	-	-	-	-	-	-	198,848	-	-	198,848	-	198,848
Exchange Rates Changes	-	-	-	-	-	(354)	-	-	-	-	(354)	-	(354)
Loss attributable to minority interests	-	-	-	-	-	-	-	-	-	-	-	299	299
Actuarial gains/losses	-	-	-	-	6,778	-	-	-	-	-	6,778	-	6,778
Income and expenses recognised in the period	-	-	-	-	6,778	(354)	-	198,848	-	-	205,272	299	205,571
Transactions with Shareholders:													
Distribution of profit:													
To reserves	-	-	113,073	-	(102,684)	-	-	(10,369)	-	-	-	-	-
To dividends (Note 14 e)	-	-	-	-	-	-	-	(163,572)	54,116	-	(109,456)	-	(109,456)
Dividends (Note 14 e)	-	-	-	-	-	-	-	-	(56,714)	-	(56,714)	-	(56,714)
On treasury shares operations (Note 14 b and 14 f)	-	-	718	-	-	-	-	-	-	(3,917)	(3,199)	-	(3,199)
Incentive Plan (Note 4.12)	-	-	768	-	-	-	-	-	-	1,863	2,431	-	2,431
Business combinations	-	-	-	-	-	-	-	-	-	-	-	3,691	3,691
Others	-	-	-	-	2	-	-	-	-	-	2	(105)	(103)
BALANCE AS OF SEPTEMBER 30, 2022	26,550	867,808	403,573	(753,349)	(127,639)	(451)	19,950	198,848	(56,714)	(16,600)	561,676	4,719	566,695
Net profit attributable to the Parent	-	-	-	-	-	-	-	272,254	-	-	272,254	-	272,254
Exchange Rates Changes	-	-	-	-	-	107	-	-	-	-	107	-	107
Loss attributable to minority interests	-	-	-	-	-	-	-	-	-	-	-	410	410
Actuarial gains/losses	-	-	-	-	2,530	-	-	-	-	-	2,530	-	2,530
Income and expenses recognised in the period	-	-	-	-	2,530	107	-	272,254	-	-	274,891	410	275,301
Transactions with Shareholders:													
Distribution of profit:													
To reserves	-	-	6,794	-	10,864	-	-	(16,878)	-	-	-	-	-
To dividends (Note 14 e)	-	-	-	-	-	-	-	(181,970)	56,714	-	(125,256)	(36)	(125,262)
Dividends (Note 14 e)	-	-	-	-	-	-	-	-	(64,619)	-	(64,619)	-	(64,619)
On treasury shares operations (Note 14 b and 14 f)	-	-	4,804	-	-	-	-	-	-	(7,332)	(2,526)	-	(2,526)
Incentive Plan (Note 4.12)	-	-	(118)	-	-	-	-	-	-	2,667	2,549	-	2,549
Business combinations (Note 7.2)	-	-	(56,223)	-	-	-	-	-	-	-	(56,223)	(488)	(56,712)
Others	-	-	-	-	(83)	-	-	-	-	-	(83)	-	(83)
BALANCE AS OF SEPTEMBER 30, 2023	26,550	867,808	358,830	(753,349)	(115,106)	(344)	19,950	272,254	(64,619)	(21,265)	590,707	4,804	595,311

Figura 4.9: Tabla procesada con LlamaParse [32] en modo avanzado (formato *markdown*)

los documentos, se enviarán a Llama Parse para el procesamiento, obteniendo el texto formateado en *markdown*.

De ahora en adelante, LlamaParse será utilizado como una etapa de preprocesamiento dentro del proceso de creación de la base de datos vectorizada para el RAG. Nada más los archivos hayan sido subidos a la web, el primer paso será procesarlos con LlamaParse y después continuar con la etapa de chunking y demás pasos hasta la creación de la base de datos.

4.2. Otras técnicas implementadas

De cara a perfeccionar los agentes y flujos de trabajo, es importante detenerse en cada etapa e intentar mejorarla. Por ello, se evalúan distintas técnicas de vanguardia para perfeccionar el pipeline de búsqueda local.

4.2.1. Chunking

El RAG básico separa el texto en pedazos de igual longitud, sin tener en cuenta nada más que el número de caracteres establecido. Esto trae muchos problemas, y

lo único bueno es la facilidad computacional. El principal error que trae es que no distingue por dónde se corta el texto. Es decir, que a pesar de haber realizado un preprocesado que traduce a *markdown* las tablas, estas podrían verse cortadas a la mitad (perdiendo todo su valor). Y no solo tablas, sino que los párrafos, oraciones, títulos, todo ello es vulnerable de ser separado en dos por el chunking.

Por ello, se ha implementado un chunking personalizado que:

- Mantiene la estructura de secciones:
 - Cada encabezado markdown (#) define el inicio de una sección.
 - Se agrupa todo el contenido bajo el mismo encabezado hasta el siguiente.
- Respeta bloques indivisibles dentro de cada sección:
 - *Tablas completas*: líneas que comienzan con | se agrupan sin romperse.
 - *Párrafos*: texto separado por líneas vacías se trata como unidad única.
- Controla el tamaño de cada chunk para aproximarse a ≈ 1000 caracteres, con tolerancia de ± 150 :
 - Se añaden bloques consecutivos (tabla o párrafo) mientras el total no supere 1150 caracteres.
 - Si al incorporar un bloque aún estamos por debajo de 850 caracteres, lo incluimos para no generar fragments demasiado pequeños.
 - En caso contrario, se cierra el chunk actual y se comienza uno nuevo, siempre sin cortar bloques.

De este modo, cada fragmento resultante contiene unidades de sentido completo, evita la pérdida de información en la división y mantiene las tablas y párrafos íntegros, optimizando al máximo el tamaño de los chunks para su posterior vectorización.

4.2.2. Hybrid Search

Hasta el momento, en la etapa de retrieval o de obtención de los pedazos de texto, siempre se ha usado búsqueda por similitud. Sin embargo, esto hace que el sistema dependa exclusivamente de este método. Concretamente, si la petición del usuario busca datos muy específicos.

Los componentes de la búsqueda híbrida son los siguientes:

- **Búsqueda semántica (embeddings):** Aprovecha la base de datos vectorial (ChromaDB) para recuperar fragmentos cuyo embedding sea cercano al de la consulta, capturando coincidencias de significado más allá de la mera ocurrencia de términos (enfoque anterior).
- **Búsqueda por palabras clave (BM25):** Emplea un índice invertido y el algoritmo BM25 para detectar fragmentos que contengan explícitamente los términos de la consulta, garantizando precisión en coincidencias literales.

El flujo de implementación que sigue es muy similar al anterior pero la etapa del retrieval es la única realmente diferente. A continuación se detallan los pasos:

1. **Introducción y contexto:** Se define el concepto de búsqueda híbrida y se justifica su uso frente a la búsqueda puramente semántica.
2. **Procesamiento de documentos:** Los archivos Markdown se extraen de un directorio, se fragmentan (chunking) y se preparan para la indexación.
3. **Creación y carga de la base vectorial:** Se genera o carga una instancia de ChromaDB que almacena los embeddings de cada fragmento.
4. **Implementación de la búsqueda híbrida:**
 - Se lanza la consulta al módulo de embeddings y al índice BM25.
 - Se combinan ambos conjuntos de resultados, eliminando duplicados.
 - Se prioriza la diversidad de fuentes para enriquecer el contexto.
5. **Integración con RAG:** El resultado de la búsqueda híbrida alimenta al modelo de lenguaje (p. ej. GPT-4o-mini), que genera respuestas fundamentadas.

Con esta arquitectura de búsqueda híbrida, el sistema deja de depender únicamente de la similitud de embeddings y mejora tanto la precisión en términos literales como la capacidad de comprender consultas complejas. Sin embargo, aunque la teoría indica que podría ser útil, como analizamos en el capítulo de resultados 6, no se ve clara consistencia y suficiente mejora como para implementar esta técnica. Aumentar el número de chunks a obtener de la base de datos incrementa el costo de cada operación y al no haber suficiente mejora se decide no implementar esta técnica.

4.3. RAG Agent

Tras explorar el RAG básico, implementar un filtrado por metadatos e incluir una etapa de preprocesamiento, todavía hay mucho que optimizar en cuanto a arquitectura global y flujo de trabajo. Se busca desarrollar un agente con acceso al pipeline de RAG, que tenga cierta autonomía de acceder a la base de datos las veces que sea necesario.

Esta autonomía implica que el agente puede trabajar por su cuenta dentro de un flujo de trabajo. Esto permite que acceda repetidas veces a una base de datos. También, se busca incluir varios salvoconductos para garantizar la eficacia del RAG y de las respuestas generadas.

Además, como esto es una herramienta para los analistas, se pretende reducir la fricción para su uso. Por petición de los analistas de Azvalor, la creación de la base de datos en tiempo real es una funcionalidad que se valora desarrollar.

4.3.1. Objetivos

Entonces, los objetivos de esta iteración de la funcionalidad local son los siguientes:

- Arquitectura RAG: diseñar un agente con recursividad y bucles de verificación, incluyendo las técnicas más eficientes para la búsqueda local y desarrollo de agentes IA.
- Creación de Base de Datos Automatizada: implementar un pipeline automatizado para subir los documentos de una empresa y que en segundos se genere una base de datos con la que trabaje el LLM.
- Técnicas del estado del arte: incorporar técnicas validadas y publicadas recientemente para obtener un agente de vanguardia.
- Estado del agente en tiempo real: mostrar el proceso que va realizando el agente en vivo, para que el usuario pueda saber en qué etapa se encuentra su petición.
- Reducir alineaciones: diseñar una arquitectura que reduzca el porcentaje de los errores del modelo y la falta de contexto relevante del RAG, reduciendo los errores de este pipeline a su vez.

Con ello se busca crear un agente que asista a los analistas para generar evoluciones de métricas financieras a lo largo de los años, extraer información de varios documentos y analizar transcripciones.

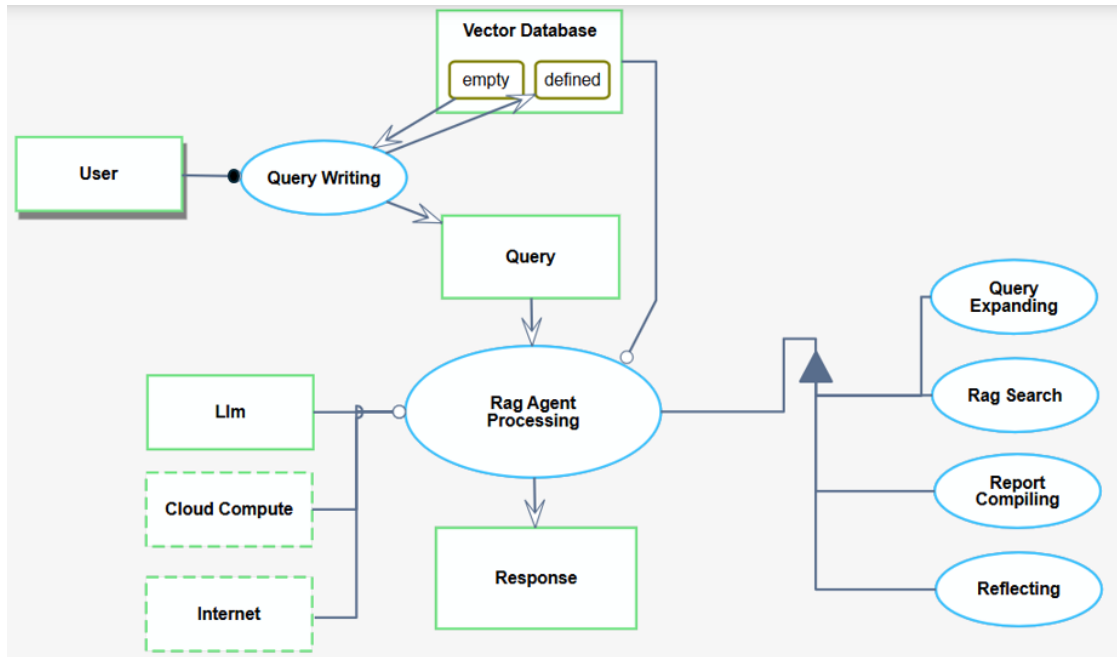


Figura 4.10: Esquema OPM: Arquitectura del Agente RAG

4.3.2. Descripción de la Arquitectura

Para definir el diseño de este agente se priorizan los objetivos previamente mencionados. La estructura del agente está enfocada en reducir alucinaciones y obtener extractos lo más relevantes posibles. El agente presentado extiende el esquema clásico de RAG mediante un *grafo de estados* con cinco nodos principales:

1. **Expansión de consulta:** Genera y refina consultas usando plantillas de prompt y llamadas estructuradas (`function_calling`).
2. **Recuperación:** Busca documentos relevantes en un `vectorstore` (Chroma + OpenAIEmbeddings) según las consultas.
3. **Reflexión sobre documentos:** Filtra documentos relevantes y genera pensamientos sobre información faltante, similar al mecanismo de reflexión de Self-RAG.
4. **Generación del informe:** Produce un reporte fundamentado en los documentos filtrados.
5. **Reflexión de completitud:** Evalúa si el informe cubre totalmente la pregunta; de no ser así, vuelve a la fase de expansión de consulta y centra la búsqueda en los aspectos ausentes.

Este flujo iterativo controla cuándo detener la recuperación y la generación, evitando tanto la sobrecarga de documentos como respuestas incompletas.

Inspiración en Self-RAG y Adaptive-RAG

El Self-RAG [4] introduce la idea de que el modelo reflexione durante la generación, marcando cuándo recuperar o criticar su propio output. Aunque aquí no se usan tokens especiales, el agente emplea nodos de reflexión y salidas estructuradas (`ReflectionDocs`, `ReflectionCompleteness`) para guiar el flujo de ejecución de forma análoga.

El Adaptive-RAG [27] propone un mecanismo dinámico para decidir cuándo recuperar información basándose en la complejidad de la consulta o la incertidumbre del modelo. En el agente aquí desarrollado se implementan las siguientes características:

- El agente ajusta el número de iteraciones de recuperación (`MAX_ITERATIONS_RETRIEVAL`) según la presencia de documentos relevantes.
- Detiene la fase de recuperación cuando se cumplen criterios de completitud, similar a detectores de punto óptimo de recuperación en Adaptive-RAG.

4.3.3. Especificaciones Técnicas

Para implementar el Agente RAG se han utilizado las siguientes especificaciones y componentes:

- **Lenguaje y entorno:** Python 3.9+, con dependencias principales en `langchain`, `langgraph`, `pydantic` y `chromadb`.
- **Modelo de lenguaje:** ChatOpenAI basado en GPT-4o, configurado con `temperature=0` para máxima determinismo y `max_tokens=3000`.
- **Embeddings:** OpenAIEmbeddings (modelo `text-embedding-3-small`), usado para vectorizar documentos.
- **Almacén vectorial:** Chroma con persistencia en `./vectorstore_chromadb`, realizando búsquedas de similitud (`k=2` por consulta, justificado en el capítulo 6).
- **Pipeline y orquestación:**
 - Grafo de estados (`StateGraph`) con nodos: `query_expansion`, `retrieval`, `reflection_docs`, `generation` y `reflection_completeness`.
 - Flujo recursivo que permite hasta `MAX_ITERATIONS_RETRIEVAL=2` iteraciones de búsqueda y `MAX_ITERATIONS_REFLECTION=2` iteraciones de reflexión.
 - Uso de `with_structured_output` para salida tipada (Pydantic) en las fases de expansión de consulta y reflexión.

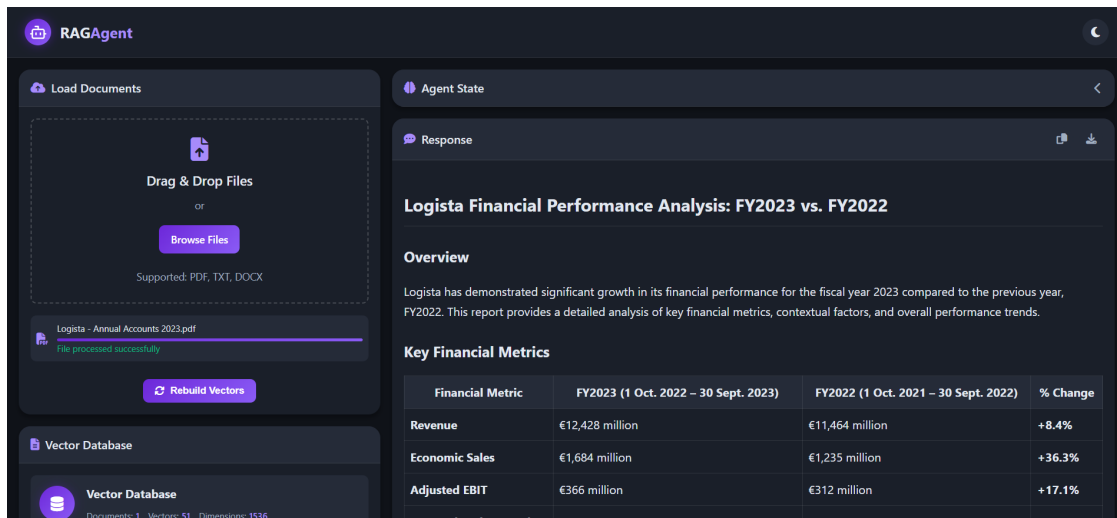


Figura 4.11: Interfaz del Agente RAG

- **Monitorización en tiempo real:** Cada nodo emite mensajes con un `custom_key` que se transmiten al usuario, permitiendo visualizar en vivo el estado (“Planning...”, “Retrieving...”, “Generating report...”, etc.).

4.3.4. Resultados

En el anexo C, se puede ver al Agente RAG en un ejemplo de comparativa de evolución. Se suben los archivos pertinentes, en este caso, un archivo de las cuentas anuales de una empresa. A continuación, se genera automáticamente todo el pipeline de creación de la base de datos. Y, por último, el usuario lanza su petición. Puesto que se le pide un análisis de los resultados financieros de la empresa, el agente devuelve una comparativa con las principales métricas al usuario en forma de tabla y también textual. Como se ha descrito anteriormente, se puede seguir el funcionamiento del agente en tiempo real viendo en qué estado se encuentra. El resultado es muy positivo, contesta con lenguaje financiero y con métricas específicas mostrando la evolución de los últimos años de la empresa.

4.3.5. Conclusión

En esta iteración se ha desarrollado un Agente RAG altamente autónomo y transparente que responde de manera efectiva a los objetivos planteados. Gracias a su arquitectura recursiva y modular, basada en un grafo de estados, el sistema incorpora bucles de verificación y refinamiento que aseguran una recuperación de información exhaustiva. La integración con ChromaDB y un pipeline de em-

beddings permite crear la base de datos en segundos a partir de los documentos subidos, mientras que la adopción de prompts estructurados y reflexiones automáticas sitúa al agente en la vanguardia de las mejores prácticas. La monitorización en tiempo real, mediante mensajes en consola, ofrece al analista un seguimiento detallado de cada fase del proceso, reduciendo la fricción y mejorando la trazabilidad. A su vez, los ciclos de reflexión sobre documentos y de comprobación de completitud minimizan las omisiones de contexto y garantizan respuestas completas y precisas.

Como resultado, los analistas disponen de una herramienta robusta para generar evoluciones de métricas financieras, extraer insights de múltiples documentos y procesar transcripciones de forma autónoma. En las próximas fases se trabajará en optimizar la latencia de las consultas y en ampliar el número de fuentes integradas en la base de datos automatizada.

Capítulo 5

Funcionalidad Búsqueda

Durante el análisis de empresas, no solo se trabaja con documentos emitidos por la empresa a analizar o por auditorías, sino que también se utilizan herramientas de búsqueda en línea. Mucha información es obtenida mediante otras fuentes, como puede ser la información de los dirigentes, datos del mercado y sus competidores, historia de la compañía, etc.

Es por ello que una de las funcionalidades que se desarrollan en este proyecto es la de búsqueda. De cara a complementar la base de conocimiento, se conectan los LLM con motores de búsqueda. Puesto que el objetivo es el de desarrollar el workflow idóneo implementando búsqueda, inicialmente, se incorpora la API de Tavily Search [17] que facilitará las búsquedas.

5.0.1. Tavily Search API

Tavily es una herramienta de software diseñada para la ejecución con LLM. Permite que los modelos de lenguaje hagan búsquedas y obtengan resultados de una forma ágil y sencilla. Además, facilita su implementación, proporcionando distintos parámetros y formas de búsqueda. En este caso, se emplea la opción de búsqueda simple sin procesamiento; los detalles de la parametrización son especificados en cada desarrollo.

5.1. Agente de Búsqueda 1.0

Como novedad de la empresa OpenAI, en febrero de 2025 se publicó la funcionalidad *Chat GPT Search* [38] a todos los públicos, buscando dotar a sus modelos de información actual, permitiéndole realizar búsquedas. Sin embargo, esta función se ve limitada por las siguientes razones:

- Incapacidad de realizar varias búsquedas en una misma petición.

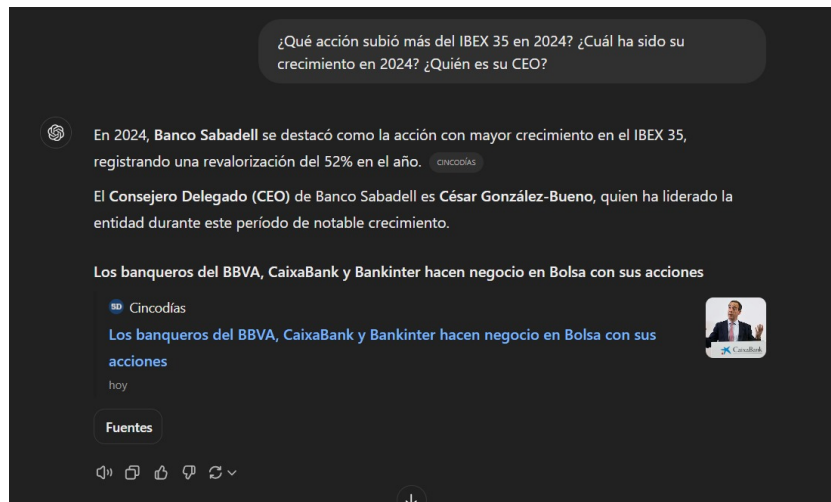


Figura 5.1: Respuesta de ChatGPT Search a petición compleja.

- Dependencia en las fuentes de búsqueda.
- Ausencia de verificaciones.

Tras diversas pruebas, como se puede observar en la figura 5.1, ChatGPT Search no es capaz de responder peticiones que requieren varias etapas de búsqueda. En este caso, sí que consigue responder a todas las preguntas de la petición; no obstante, al hacer una búsqueda simplificada (solo extrae una parte de la fuente), se equivoca, puesto que IAG fue la empresa que más subió en 2024 en el IBEX35. Este resultado es sorprendente ya que la respuesta a la pregunta se puede obtener en esa misma fuente. Sin embargo, se ve cómo esta funcionalidad está limitada y puede mejorarse para responder peticiones complejas, en este caso, se formula con varias preguntas pero, podría mostrarse como una petición compleja que requiere de información previa.

5.1.1. Arquitectura y funcionamiento

Por ello, en esta primera iteración se implementa una arquitectura siguiendo la filosofía ReAct [51]. Los agentes ReAct razonan, con etapas de planificación, decisión o reflexión y además actúan, con herramientas a su alcance. En otras palabras, esto consiste en dotar al modelo de poder de decisión y de autonomía, consiguiendo una mayor profundidad y precisión en las respuestas. No solo se implementa el motor de búsqueda, sino que también se delega en el modelo la capacidad de decidir cuándo se ha terminado de realizar búsquedas.

Entonces, si la petición requiere de distintas búsquedas secuenciales, el modelo

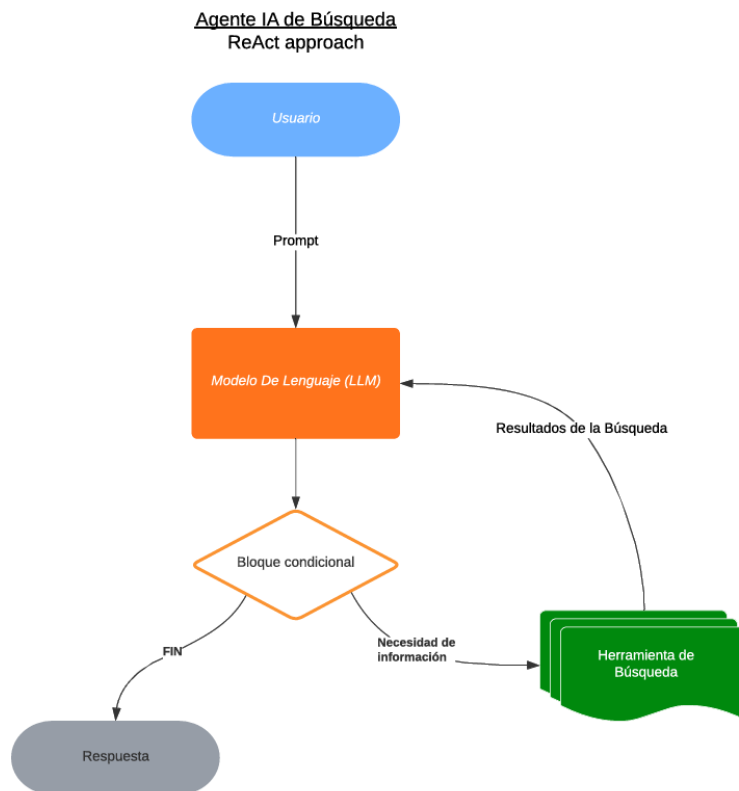


Figura 5.2: Diagrama de flujo del Agente de Búsqueda 1.0

podrá hacerlo hasta que considere que es suficiente para responder a la pregunta y, solo entonces, responde al usuario.

En definitiva, se define a este desarrollo como agente de búsqueda 1.0. Como destaca la figura 5.2, este agente está compuesto por el modelo de lenguaje como núcleo, una herramienta de búsqueda y un bloque de decisión, en donde el LLM decide si realizar una búsqueda más o terminar el bucle y responder al usuario.

5.1.2. Especificaciones

Como se puede apreciar en el código compartido, se sigue utilizando gpt-4o-mini para el desarrollo de estos agentes. Además, Tavily Search se usa en modo avanzado para mayor profundidad y mejores resultados. En concreto, se especifica el número de resultados a un máximo de cuatro, ya que tras realizar varias pruebas se observó que a partir del quinto resultado, la información obtenida empezaba a ser irrelevante (obtenido por la métrica de relevancia que concede el software).

Agente IA de Búsqueda

¿Cómo puedo asistirte hoy?

¿Qué acción subió más del IBEX 35 en 2024? ¿Cuál ha sido su crecimiento en 2024? ¿C

Buscar

Respuesta:

En 2024, la acción que más subió en el IBEX 35 fue **IAG (International Airlines Group)**, con una revalorización del **103,76%**. El índice cerró el año con un aumento general del **14,78%**, alcanzando los **11,595 puntos**.

Respecto al CEO de IAG, la empresa está liderada por **Luis Gallego**.

Para más detalles puedes consultar las siguientes fuentes:

IAG y su revalorización en 2024

Cierre del IBEX 35 en 2024

Figura 5.3: Respuesta del Agente de Búsqueda 1.0

5.1.3. Resultados

Para comparar la mejora que proporciona este primer agente, la figura 5.3 refleja como este agente es capaz de iterar búsquedas y devolver una respuesta completa a la misma pregunta hecha previamente a ChatGPT Search en la figura 5.1. La respuesta es precisa, responde a las tres partes de la petición y es capaz de devolver las fuentes en las que basa la respuesta con su correspondiente url.

5.1.4. Conclusiones

Se demuestra como utilizar un agente permite al usuario desentenderse un poco más del proceso, enviar peticiones más complejas y confiar en que el agente se encargará de realizar las búsquedas necesarias para hallar la información requerida para responder a la pregunta.

No obstante, sigue siendo un agente genérico, presenta una mejora respecto a un buscador convencional, pero no está afinado al campo financiero ni del análisis de empresas. Además, aunque presenta un poco de autonomía, podría aportar más con más libertad de búsqueda, buscando más veces en distintos ámbitos.

5.2. Agente de Búsqueda 2.0

Con el propósito de aumentar el grado de libertad y conseguir más de los agentes, se decide desarrollar un agente capaz de realizar informes de empresas. Estos informes tendrán como fin dar una vista general de la empresa de forma que, cuando un analista proceda a investigar una compañía a priori desconocida, pueda hacerse una idea de los principales aspectos de la misma. Para obtener esta descripción a alto nivel de la empresa, se revisan los siguientes apartados:

- Historia e hitos más importantes.
- Negocio: entender qué hace y toda su cadena de valor (analizar proveedores, clientes, competidores, regulador, empleados, etc.), posibles barreras de entrada o ventajas competitivas.
- Mercado: con quien compite y como son sus cuotas de mercado (y cómo han evolucionado históricamente).
- People: consejo de administración, management y accionistas. Quienes son, track record, etc. Del equipo directivo cuanto cobran y de qué dependen sus incentivos.
- Capital allocation: análisis de M&A (cantidad, múltiplos, etc.), dinero destinado a dividendos, recompras o emisión de acciones (a qué precio lo hicieron).

Con esto, se permite mejorar la eficiencia y rapidez del inicio del análisis, dando una vista de alto nivel de cualquier empresa al analista.

5.2.1. Arquitectura y funcionamiento

La segunda versión del agente de búsqueda buscará información para completar el esquema definido según los apartados anteriores. Como se describe en el esquema de la figura 5.4, el agente tendrá un workflow compuesto por los siguientes nodos:

1. **Query Generation:** se generan las queries de búsqueda para encontrar la información del esquema. En la primera iteración, se generan queries para todos los apartados y, en las siguientes interacciones, para los apartados que falten, según el feedback del nodo *Reflection*.
2. **Search:** todas estas queries son enviadas y se obtienen diversos resultados para cada una de ellas. Todos estos resultados son procesados y resumizados, extrayendo lo exclusivamente necesario para el análisis.

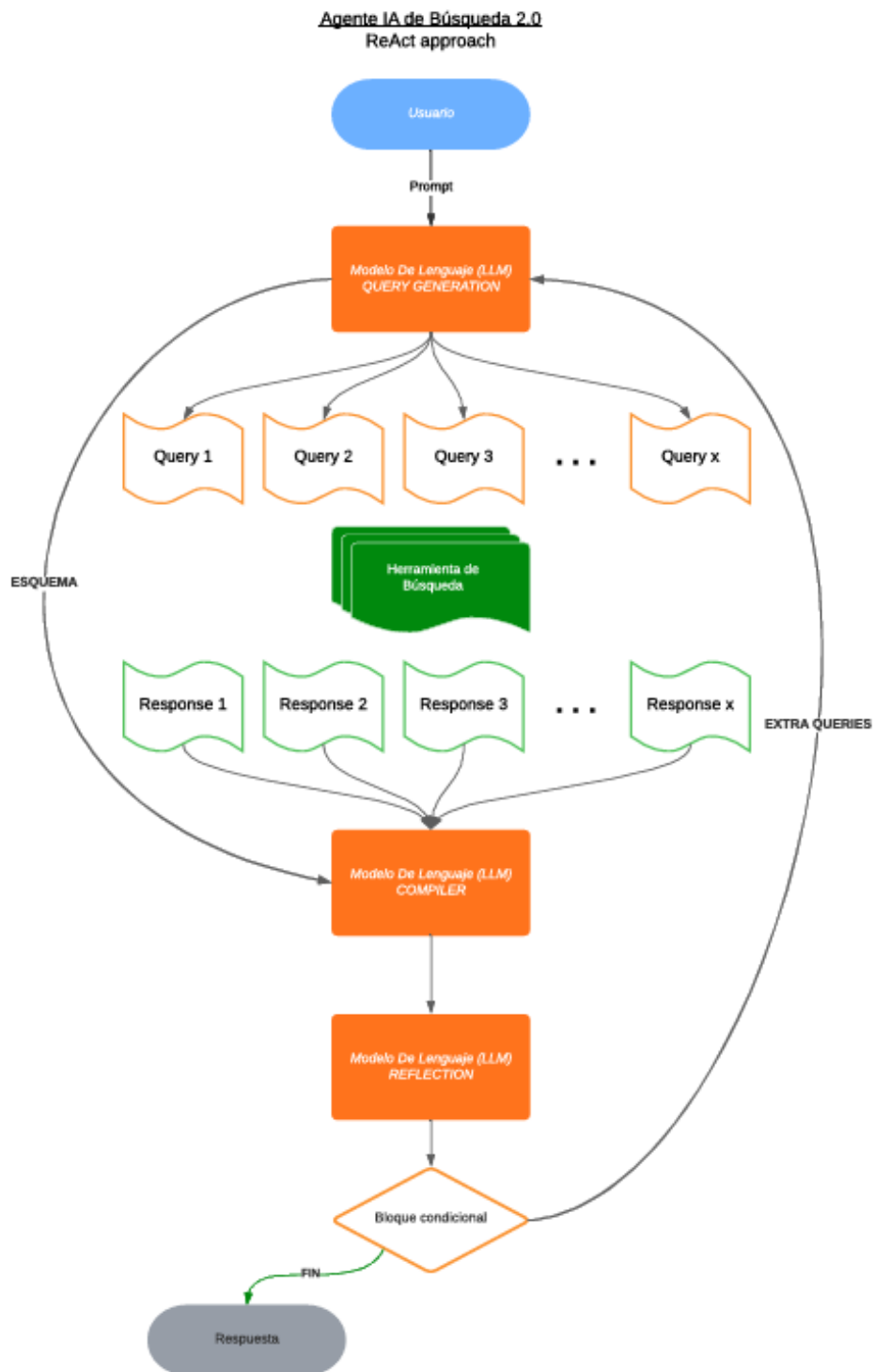


Figura 5.4: Diagrama de flujo del Agente de Búsqueda 2.0

3. **Compilation:** A partir de la información extraída, se redacta el informe final. Es decir, inicialmente el informe se realiza de cero y, en las siguientes iteraciones se completa del análisis previo.
4. **Reflection:** Este nodo es crítico, analiza el informe realizado hasta el momento y decide si está suficientemente completo o falta algún apartado por complementar. En el caso de que decida que el informe está incompleto, el agente especificara qué apartados se deben completar y qué información está ausente en concreto. Entonces, enviará este feedback al bloque de *Query Generation*, empezando una nueva iteración del workflow.
5. **Decisión:** Bloque condicional en el que según lo que el agente haya decidido en el bloque, se vuelve al nodo de búsqueda o se devuelve el informe si se ha decidido que ya está completo. Además, para evitar que el agente esté buscando continuamente, como se puede estimar que en algún caso no se halle la información de algún apartado, por falta de información en la web, se limita el número de iteraciones.

Con este pipeline, el agente se encargará de realizar un informe completo de una empresa, profundizando en los apartados descritos en el esquema proporcionado de la compañía elegida.

5.2.2. Especificaciones

En cuanto al modelo usado para llevar a cabo este agente, se mantiene el uso del gpt-4o-mini como modelo habitual. De esta forma, se pueden comparar más fácilmente con el anterior y poder atribuir el cambio enteramente al sistema desarrollado.

Especificaciones Técnicas

Para el Web Search Agent se han empleado las siguientes especificaciones y componentes:

- **Lenguaje y entorno:** Python 3.9+, con gestión de variables de entorno mediante *python-dotenv*.
- **Modelo de lenguaje:** *ChatOpenAI* basado en GPT-4o-mini, configurado con *temperature=0* para determinismo y *max_tokens=10000*.
- **Cliente de búsqueda web:** *TavilyClient* (API key en *TAVILY_API_KEY*), usando la herramienta *TavilySearchResults* para obtener hasta *max search results = 3* resultados por consulta.

■ **Grafo de estados (*StateGraph*):**

- *Nodos*: *gen_query*, *buscar*, *compilador*, *reflection*.
- *Aristas*: Flujo secuencial de generación de queries → búsqueda → compilación → reflexión, con bucle condicional hasta *max_iteraciones=1*.
- Uso de *with_structured_output* para salida tipada (Pydantic) en la generación de queries y reflexión.

■ **Manejo de estados:**

- *AgentState*: registra *company*, *pending_sections*, *queries*, *search_results*, *info_compilada*, *is_complete*, *iteraciones*.
- Ciclo de trabajo que actualiza *pending_sections* y *user_notes* según reflexión.

- **Monitorización en tiempo real:** Cada nodo emite mensajes con *writer({"custom_key": ...})* para el seguimiento del usuario en vivo y, al finalizar, con *writer({"final_key": ...})*.

La herramienta de búsqueda (Tavily) sigue siendo usada con los mismos parámetros. Aunque, como se ha descrito anteriormente, se generan múltiples búsquedas, en esta iteración síncronas. Como regla general, se estableció, como parámetros de máximos, que el agente no podía realizar más de dos iteraciones completas, tres búsquedas por query y seis resultados a contrastar. Más complejidad y parámetros más elevados llevan a mayor latencia, mayor redundancia y de fuentes con menos relevancia, sin añadir mejores resultados.

5.2.3. Resultados

Un ejemplo de uso de este agente se puede observar siguiendo el vínculo del anexo C. En el vídeo, se realiza el screening de vLex, una empresa LegalTech. Con tan solo introducir el nombre de dicha empresa, el agente comienza su flujo de búsqueda. En la pestaña de *Agent State*, aparece el estado en el que se encuentra el agente de forma dinámica dentro del proceso. De esta forma, se puede seguir lo que realiza en tiempo real el agente y apreciar las iteraciones que toma para completar la respuesta.

Cuando el agente finaliza el procesado, devuelve el informe final, renderizado por el frontend en formato *markdown* para preservar el texto plano junto con las tablas, preservando la estructura del documento inicial.

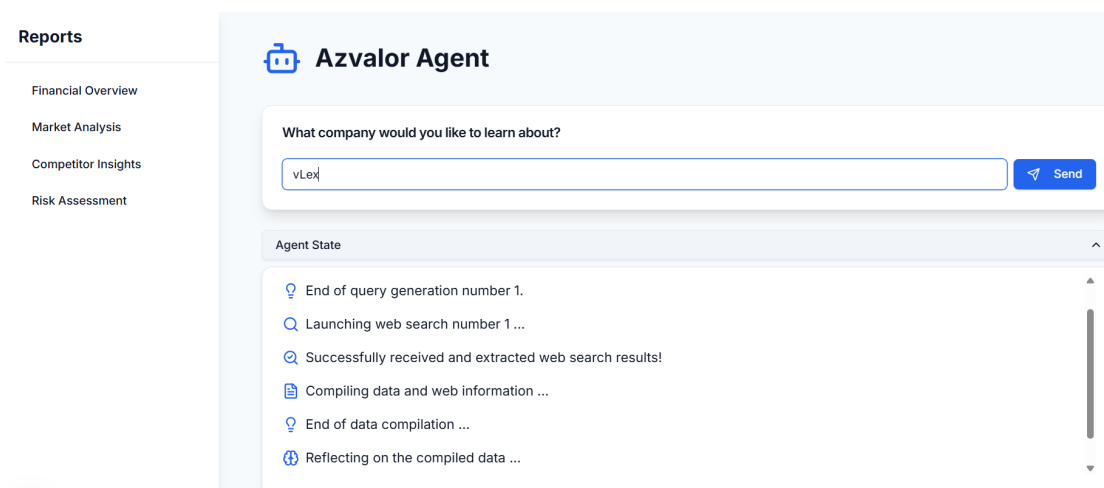


Figura 5.5: Interfaz del Agente de Búsqueda Web

5.2.4. Conclusión

El agente de búsqueda 2.0 genera informes que, aunque pretendan ofrecer una visión de alto nivel de la empresa, son capaces de exponer datos concretos, información actualizada y gran profundidad en el análisis. Todo ello, permitiendo al analista una mayor facilidad al iniciar el proceso de análisis de un negocio.

Por otro lado, este agente requiere de una mayor complejidad computacional y, debido a las numerosas búsquedas junto con las distintas llamadas a los LLM, suele demorarse en torno a dos minutos en proporcionar el informe. Esto sigue siendo aceptable puesto que este informe sería algo que, de forma manual, una persona tardaría mucho más tiempo.

Para reducir la latencia, existen aspectos a mejorar, como el paralelismo de las búsquedas. De cara a producción, estos tiempos se podrían mejorar con modelos más eficientes desplegados en local o en la nube. Además, tan solo permite peticiones individuales, sin posibilidad de mantener una conversación con el agente. Esto podría ser útil cuando el usuario tiene dudas puntuales y necesita aclaraciones o mayor profundidad de la búsqueda en algún punto.

Por último, el esquema está hardcodeado en el agente. En otras palabras, no es posible cambiarlo de forma dinámica en cada petición y el agente siempre sigue el mismo esquema. Esto reduce el uso de la plataforma solo al screening de empresas, pudiendo ser una herramienta algo más flexible dentro del sector financiero.

5.3. Global Agent

Descripción de la Arquitectura

En los capítulos anteriores se describieron los dos pilares de la plataforma. La funcionalidad local, con el agente RAG especializado en Retrieval-Augmented Generation sobre fuentes internas (10-K, transcripciones de conferencias, informes de analistas, notas internas, etc.). La funcionalidad externa, con el agente de búsqueda web orientado a enriquecer el análisis con información pública y actualizada que permite analizar perfiles de directivos, notas de prensa, artículos especializados, etc.

Aunque ambos agentes ofrecen gran valor por separado, no se unifican y complementan de forma conjunta. Esto puede llevar a redundancias y a tiempo por parte de los analistas para revisar y contrastar la información. En otras palabras, el analista tendría que pasar primero por uno, luego por otro y a partir de ahí elaborar un informe unificado de forma manual. Además, el agente de búsqueda es bastante rígido en cuanto al esquema, no permite más que analizar empresas, con un esquema preinsertado. Y, por otra parte, el Agente RAG no presenta el filtrado por metadatos desarrollado en la sección 4.1.

Para cubrir este hueco se diseña el Agente Global, cuyo cometido es orquestar y optimizar la colaboración entre los módulos existentes y los que puedan añadirse en el futuro.

5.3.1. Objetivos del Agente Global

- Calidad de la respuesta: combinar en una única salida la evidencia procedente del Agente RAG y del Agente Web, priorizando la trazabilidad y la consistencia.
- Flexibilidad: permitir que la estrategia de búsqueda sea dinámica, el agente global decide cuándo, cómo y con qué parámetros delegar a cada agente.
- Escalabilidad: Facilitar la incorporación de nuevos workers (p. ej. un sentiment-analysis agent o un market-signals agent) sin reescribir la lógica de orquestación.
- Memoria: incorporar memoria dentro de la conversación para que el agente pueda responder con información de la que se ha hablado o tenerla en cuenta para delegar.
- Trazabilidad y auditoría: Conservar los metadatos de cada paso (fuente, score de relevancia, fecha) para generar reportes y depurar incidencias.

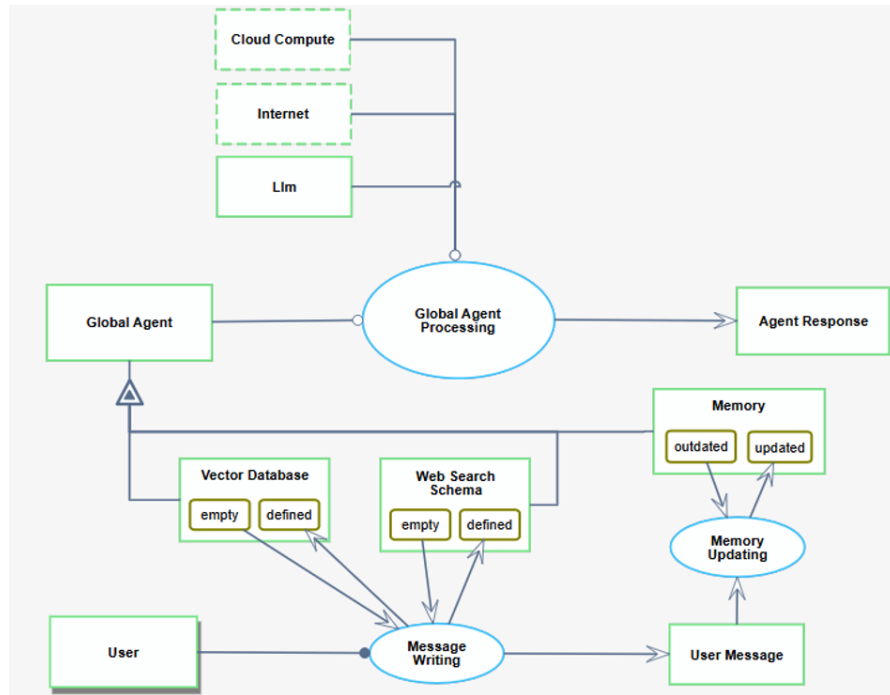


Figura 5.6: Esquema OPM: Arquitectura del Agente Global

- Razonamiento: mejorar el rendimiento general del sistema agencial con modelos de razonamiento.

5.3.2. Arquitectura y Funcionamiento

Funcionamiento alto nivel

Una de las principales funcionalidades incorporadas en este último desarrollo ha sido la de la memoria conversacional. En el diagrama OPM de la figura 5.6 se aprecia la estructura a alto nivel de este agente. El proceso principal ubicado en el centro del diagrama se detalla en el siguiente nivel del OPM en las siguientes secciones.

Esta memoria conversacional permite al agente tener constancia de anteriores mensajes. Asimismo, el usuario tiene una plataforma con fricción reducida, al tener algo más que simples preguntas y respuestas. Como indica el OPL de la figura, el proceso comienza con el mensaje del usuario, que define no solo la petición, sino también el esquema para la búsqueda en internet y la base de datos vectorizada. Estos son los inputs que se le presentan al agente global y que permiten su procesamiento habitual. Con todo ello, el agente devuelve una respuesta, a continuación se describe cómo.

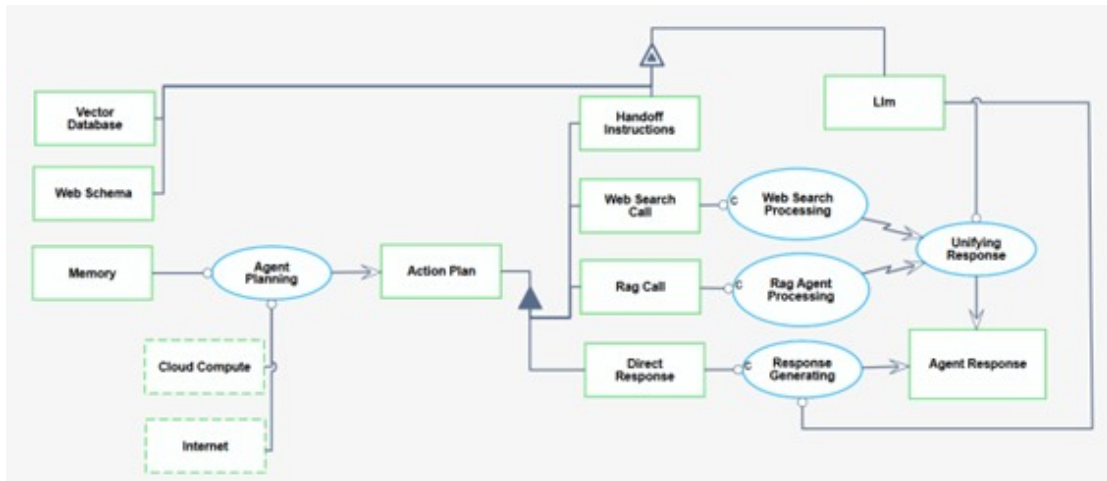


Figura 5.7: Esquema OPM: Estructura Interna del *Global Agent Processing* 5.6

Arquitectura

Como se puede observar en la figura 5.7, este agente presenta diferentes etapas y ramas. Siguiendo la metodología OPM se puede entender a alto nivel a este agente. A grandes rasgos, es un agente supervisor, que con un primer nodo de planificación, orquesta a los otros dos agentes. Este primer paso consiste en una etapa de razonamiento en la cual se interpreta tanto el último mensaje como los anteriores para determinar cuál deberían ser los siguientes pasos. En función del plan establecido, se delegan tareas con sus debidas instrucciones al Agente RAG, al Agente de Búsqueda Web, a los dos en paralelo, o a ninguno.

A continuación, se profundiza más en detalle las acciones de cada etapa:

1. Agent Planning: genera un plan de ejecución indicando qué agente consultar, con qué instrucciones y en qué orden. El input de entrada es la memoria, que presenta todos los mensajes de la conversación. El modelo llamado por API en este caso no es un LLM convencional, sino que se trata de un modelo de razonamiento.
2. Delegación a los agentes: tras la primera etapa, se pueden dar distintas posibilidades:
 - Necesidad de búsqueda local(exclusivamente): la información necesaria se encuentra en los archivos locales o el usuario especifica que solo se trabaje con los archivos subidos a la plataforma. Se habilita el Agente RAG y el propio modelo de razonamiento redacta sus instrucciones.
 - Necesidad de búsqueda externa (exclusivamente): la información necesaria para completar la petición se debe hallar en Internet por petición

del usuario o no se encuentra en los archivos. Se habilita el Agente de Búsqueda Web y el propio modelo de razonamiento genera sus instrucciones.

- Necesidad de búsqueda externa y local: la información necesaria es muy general, se puede complementar entre documentos en Internet y en los adjuntados. Entonces, los dos agentes son habilitados, con sus respectivas instrucciones. Esta ejecución se realiza en paralelo como se comentará más adelante.
- Ausencia de necesidad de habilitar más agentes: la información necesaria la dispone el modelo o se encuentra en la conversación. El modelo de razonamiento responde directamente a la petición, ya sea apoyándose en mensajes anteriores o en su propio conocimiento.

3. Unifying Response: en caso de haber llamado a todos los agentes, se necesita combinar y contrastar lo obtenido por ambos caminos. Puesto que se han activado los agentes en paralelo, se espera la respuesta de cada uno y posteriormente se juntan las respuestas en un informe final redactado por otro LLM.

5.3.3. Mejoras de los agentes anteriores

En esta plataforma final, se encuentran tres secciones: la interfaz del Agente RAG, la interfaz del Agente de Búsqueda Web, la interfaz del Agente Global. De esta forma, el usuario puede forzar una funcionalidad u otra. No solo se unifica todo en una plataforma, sino que también se han mejorado los agentes desarrollados previamente. A continuación, se detallan los distintos avances y cambios respecto a los desarrollos anteriores.

Integración del Agente RAG

Como se ha definido anteriormente, el Agente RAG, a pesar de su arquitectura depurada, no incluye el filtrado por metadatos. El filtrado en el proceso del RAG (ampliamente desarrollado en el apartado 4.1) se decide implementar en este agente de cara a que el agente pueda tener conciencia de los documentos que han sido adjuntados, que a la hora de extraer información el local el proceso sea óptimo y que se puedan hacer comparaciones de evoluciones a lo largo de los años.

En este caso, los metadatos empleados son sencillos, serán el título del documento junto con el año. Con ello habilitamos el filtrado más crucial. Esto implica que es necesario añadir una forma para que el modelo haga una query especificando los años si es útil para la petición. Además, en el pipeline de creación de

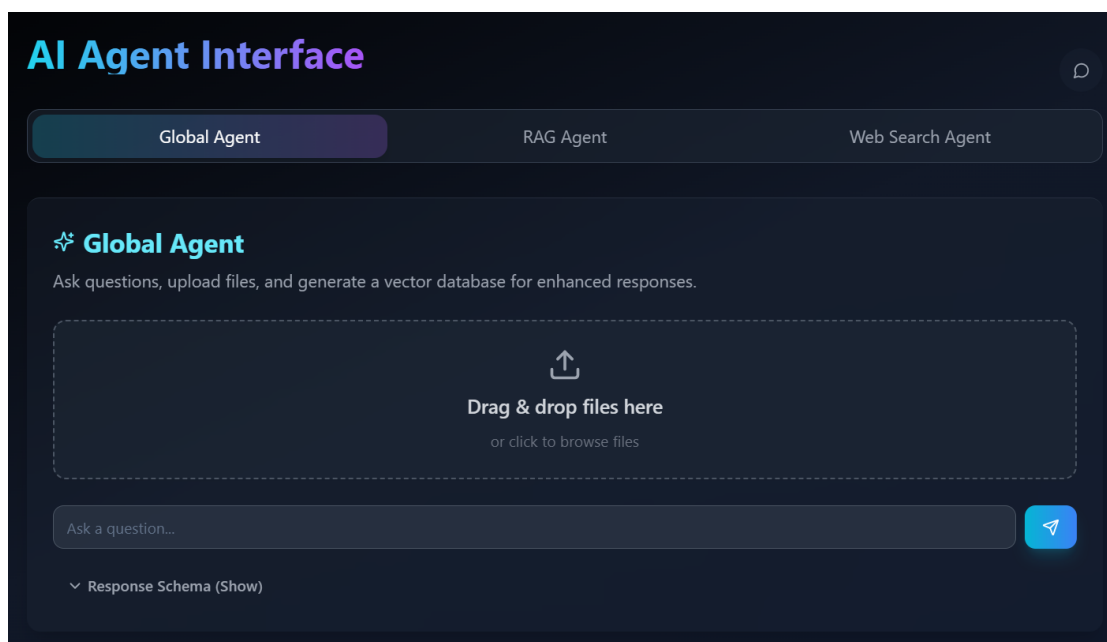


Figura 5.8: Interfaz de usuario del agente global

la base vectorizada, se deben categorizar los documentos para añadir unos metadatos. Para ello, se utiliza regex en este caso debido a la facilidad de generación de los metadatos usados. Más adelante, este proceso debería mejorarse para poder categorizar mejor los documentos y, en consecuencia, mejorar este proceso.

Integración del Agente de Búsqueda Web

Para dinamizar el uso de esta herramienta, el esquema, anteriormente introducido de forma fija, es ahora flexible a cada conversación. De esta forma, la búsqueda externa es más flexible y se puede usar para muchos más tipos de búsqueda.

Por ejemplo, ahora se pueden analizar los directivos de una empresa en profundidad. Si el analista quiere saber acerca del consejero delegado de una empresa, ahora el agente de búsqueda puede recibir una estructura que, por ejemplo, contenga su carrera, puestos pasados y estudios, y devolver estructurada una respuesta con todo detallado.

La integración consiste en que cuando el mensaje es enviado, si el usuario ha introducido un esquema o seleccionado el sugerido, este es enviado junto con el mensaje. En consecuencia, el agente actúa distinto en función de si hay esquema o no.

5.3.4. Especificaciones Técnicas

En este caso, después de la programación y debugging del agente, se implementan los modelos IA más recientes. Como LLM convencional para la mayoría de peticiones API, se usa el GPT-4.1. Este modelo tiene como virtud principal la de seguir estrictamente las instrucciones. Por otro lado, como modelo de razonamiento, se emplea el o4-mini. Este modelo de razonamiento añade cierta latencia, puesto que el modelo razona durante unos segundos, pero mejora las respuestas, sobre todo al inicio, con menos contexto.

5.3.5. Resultados

Cualitativamente, las respuestas han mejorado mucho. Al usar modelos más recientes, la calidad de las respuestas es notablemente mejor. Por un lado, los modelos dan respuestas más largas, lo cual hacía que, con modelos más antiguos y pequeños, después de recopilar mucha información, las respuestas fueran muy esqueléticas. Por otro, los modelos adecuan mejor su tono al estilo financiero, incorporando el lenguaje más naturalmente. Esto se aprecia especialmente en la delegación del agente supervisor a los otros, con unas instrucciones más completas y a la vez precisas.

En el anexo C, se puede ver cómo es una conversación con el agente global. En el vídeo incluido se observan un par de peticiones acerca de los directivos, mostrando un uso habitual de la herramienta. El agente en un caso extrae información del Consejero Delegado del documento adjuntado para exponer sus incentivos y remuneraciones. En el otro caso, busca en internet para obtener la formación y experiencia laboral previa del mismo, sin especificar su nombre, ya que con la memoria conversacional ya tiene esa información del mensaje anterior.

5.3.6. Conclusiones

La incorporación del Agente Global representa un salto cualitativo en el diseño de la plataforma, transformando un conjunto de agentes aislados en un sistema agencial cohesionado, eficiente y escalable. Su principal aportación es la capacidad de orquestar, de manera inteligente y flexible, la interacción entre distintas fuentes de información, tanto internas como externas, lo que permite generar respuestas más completas, relevantes y trazables.

Gracias a su arquitectura basada en razonamiento y memoria conversacional, el Agente Global permite al analista realizar una única consulta y recibir una respuesta integral, debidamente estructurada y jerarquizada. Esta mejora no solo reduce la fricción de uso, sino que también optimiza la productividad del analista, al minimizar tareas redundantes y ofrecer un punto de acceso unificado al

conocimiento.

Además, su diseño modular y escalable abre la puerta a futuras extensiones funcionales, como la incorporación de nuevos agentes especializados. Asimismo, la inclusión de filtrado por metadatos y esquemas flexibles mejora significativamente la calidad de la evidencia recuperada y la adaptabilidad del sistema a distintos casos de uso.

En definitiva, el Agente Global convierte la plataforma en un ecosistema inteligente de análisis empresarial, elevando la cobertura informativa sin comprometer la precisión, y posicionando la solución como una herramienta de alto valor añadido para analistas financieros y profesionales del sector.

Capítulo 6

Resultados

6.1. Testing RAG

6.1.1. Metodología para la evaluación de agentes

Evaluar los distintos pipelines tiene un grado de dificultad medio-alto ya que es difícil obtener resultados cuantitativos de modelos probabilísticos. Por ello, en la industria de la IA es habitual usar benchmarks para ver cómo de buenos son los modelos. En el caso de este proyecto, no son modelos de IA lo que se quiere evaluar sino flujos y agentes RAG.

Entonces, se emplea el mismo modelo para todos los casos, de esta forma la única diferencia será el pipeline empleado o la técnica implementada. El modelo usado es el gpt-4o-mini, puesto que se harán muchas peticiones; se elige este modelo por sus respuestas rápidas y de coste reducido.

De cara a elegir un benchmark sobre el que analizar los distintos pipelines, se priorizan distintos factores. Primero, hay que tener claro que en gran parte de este caso de uso los analistas trabajan con documentos del ámbito financiero. Por otro lado, lo que se quiere evaluar es la capacidad del retrieval y generación de respuestas. En otras palabras, se busca ver cuán bien el sistema diseñado extrae la información y responde a la pregunta.

Existen muchos benchmarks, y con el auge actual de los modelos IA, hay muchas publicaciones con datasets nuevos. El benchmark FinQA [12] es un gran ejemplo del tipo de benchmark más popular en el sector financiero. Sin embargo, este evalúa operaciones a partir de contextos muy pequeños, algo que no interesa en este proyecto ya que se analizan documentos muy largos y las operaciones no son relevantes. FinTextQA [11] también presenta pares de preguntas con sus respuestas pero no encaja del todo al tener documentos pequeños. Además, existen muchos datasets enormes con miles de preguntas y respuestas como TriviaQA [28].

Sin embargo, este tipo de benchmarks están pensados para directamente entrenar modelos y luego probarlos.

FinRepQA benchmark y RAGAS framework

Analizando las necesidades de este testing, el benchmark FinRepQA [41] es ideal para este escenario. Presenta diversos documentos 10K¹, sobre los que se basan las preguntas. Aparte de las preguntas, el benchmark muestra el contexto del documento en donde está presente la información necesaria para responder y el ground truth que es la respuesta correcta a la pregunta. Con esto, se pueden evaluar los distintos diseños y sistemas para comparar sus resultados entre sí.

No obstante, si se pretende realizar un testing en profundidad, con muchas preguntas, no es viable analizar cada respuesta y contexto extraído de forma manual. Por ende, es necesario automatizar esta evaluación para obtener los resultados de corrección. Aquí es donde el framework RAGAS [18] es de gran utilidad.

Ragas es un framework diseñado para evaluar de forma sistemática y modular aplicaciones de Retrieval-Augmented Generation (RAG). Su objetivo es facilitar un desarrollo “metrics-driven”, donde cada componente de la canalización (retrieval, generación, etc.) se mide aisladamente para identificar cuellos de botella y oportunidades de mejora.

Métricas clave Para la evaluación de cada respuesta se emplean las siguientes métricas para comparar unos agentes y otros.

- **Context Precision** Mide la proporción de fragmentos recuperados que son realmente relevantes. Se define como la precisión promedio sobre los K primeros elementos recuperados:

$$\text{Context Precision} = \frac{1}{K} \sum_{k=1}^K v_k,$$

donde

$$v_k = \begin{cases} 1, & \text{si el ítem en la posición } k \text{ es relevante,} \\ 0, & \text{en caso contrario.} \end{cases}$$

- **Context Recall** Mide la fracción de documentos relevantes totales que el sistema logra recuperar:

¹Estos son documentos que resumen la actividad económica y financiera de las empresas. Presentan resultados, con tablas habitualmente.

$$\text{Context Recall} = \frac{|\{\text{documentos relevantes recuperados}\}|}{|\{\text{documentos relevantes en la referencia}\}|}.$$

- **Faithfulness** Evalúa la consistencia factual de la respuesta generada respecto al contexto. Se extraen las afirmaciones (claims) del texto generado y se computa:

$$\text{Faithfulness} = \frac{|\{\text{claims inferibles del contexto}\}|}{|\{\text{claims totales en la respuesta}\}|}.$$

- **Answer Relevancy** Mide cuánto de pertinente es la respuesta a la pregunta original. Se generan N preguntas a partir de la respuesta y se calcula la similitud coseno media entre sus embeddings y el embedding de la pregunta original:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{i=1}^N \cos(E_{g_i}, E_o),$$

donde E_{g_i} es el embedding de la i -ésima pregunta generada y E_o es el embedding de la pregunta original.

Con estas métricas se compararán los distintos sistemas y agentes para probar su efectividad y la de técnicas a implementar.

6.1.2. Número óptimo de chunks

En este experimento se analiza el efecto del número de fragmentos de contexto (“chunks”) enviados al modelo en el pipeline de RAG. Para ello se emplea un único sistema base, correspondiente a un modelo RAG sencillo, y se varía únicamente la cantidad de chunks K asignados al generador. Sobre un mismo conjunto de preguntas se evalúan cinco métricas clave (relevancia de la respuesta, recall de contexto, precisión de contexto, fidelidad y puntuación global) para valores de $K=1, 2, 3$ y 4 .

Los resultados, ilustrados en la figura 6.1, revelan una mejora pronunciada de todas las métricas al pasar de $K=1$ a $K=2$. A partir de $K=2$, no obstante, los incrementos se atenúan: algunas métricas alcanzan su valor máximo en $K=4$, mientras que otras dejan de mejorar o experimentan ligeros descensos, lo cual indica rendimientos decrecientes. Más allá de $K=5$, no se anticipan mejoras sustanciales, pero sí un aumento innecesario del coste computacional y de la latencia.

Aplicando un criterio análogo al “método del codo” (elbow method), se determina que $K=2$ constituye el punto óptimo de compromiso entre calidad de respuesta

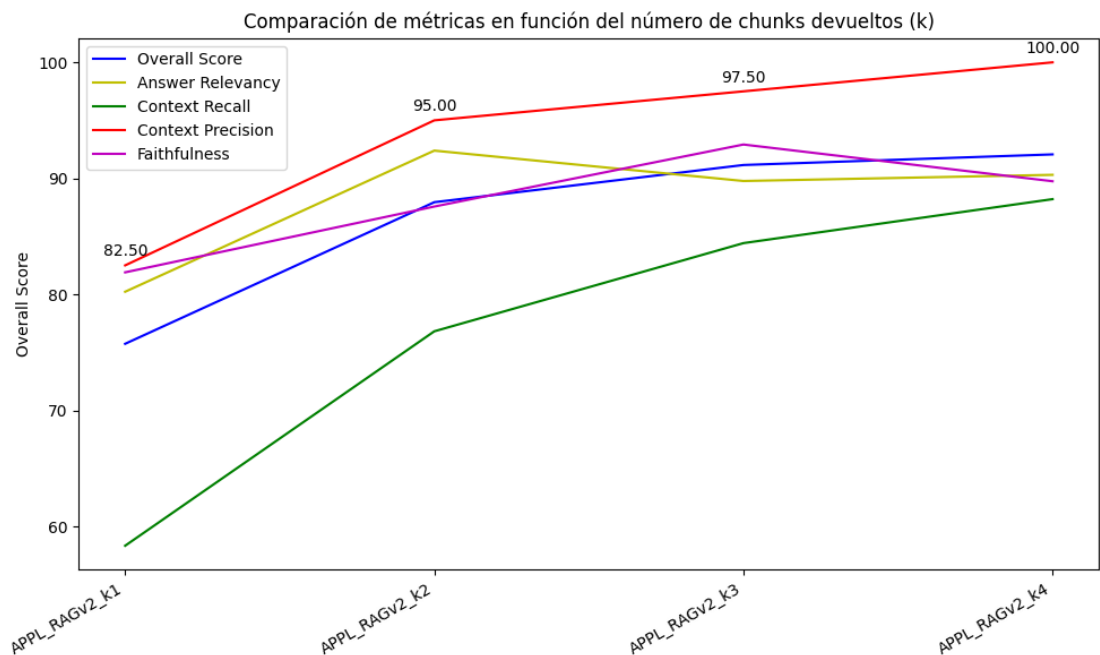


Figura 6.1: Evolución de resultados según el número de chunks

y eficiencia operativa. Con dos fragmentos de contexto, el sistema integra información suficiente para maximizar el rendimiento general, sin incurrir en redundancias ni en sobrecostes asociados al procesamiento de un número superior de fragmentos. En consecuencia, se adopta $K=2$ como configuración predeterminada para el pipeline RAG en este tipo de consultas.

6.1.3. Comparación de la precisión de los distintos sistemas y evoluciones

En este experimento se evalúa la relevancia de la respuesta sobre un conjunto de 100 preguntas, probando cinco configuraciones distintas del sistema de generación:

1. **Sin documentos** (APPL_NoDoc100s): el modelo genera respuestas sin acceso a conocimiento externo (46,03 %).
2. **Documentos completos** (APPL_NoRAG): se suministran íntegramente los textos de referencia (72,49 %).
3. **RAG básico** (APPL_RAG100): se utiliza un pipeline RAG con recuperación de los 100 fragmentos más relevantes (83,48 %).

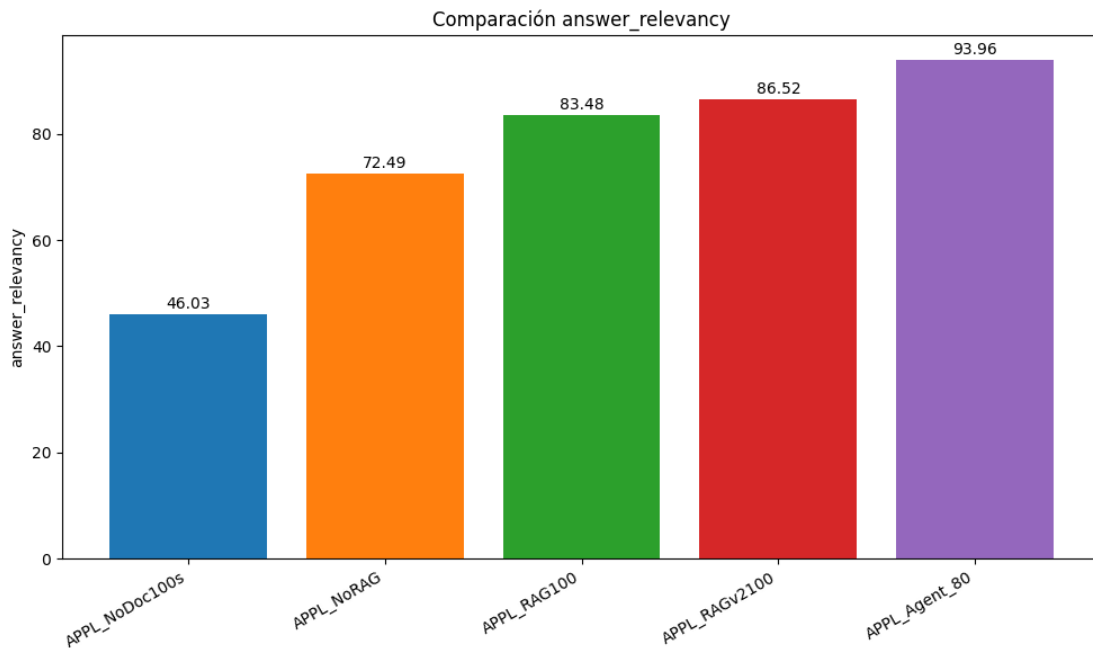


Figura 6.2: Comparativa entre los distintos sistemas de respuesta

4. **RAG con preprocesamiento de tablas** (APPL_RAGv2100):, además de la recuperación, se aplica una etapa específica para datos tabulares (86,52 %).
5. **Agentic RAG** (APPL_Agent_80'): se incorpora un componente agente que planifica iteraciones adicionales entre recuperación y generación (93,96 %).

Los resultados de la Figura 6.2 muestran un progreso continuo al incorporar mecanismos de recuperación y procesamiento de contexto. Primero, sin pasarle documentos al modelo, como son documentos de hace cierto tiempo, existen preguntas que el modelo sabe la respuesta de partida (se puede deducir que el modelo fue entrenado en textos con esta información). El acceso directo a documentos completos casi duplica la relevancia frente al escenario sin contexto. Pero, con el pipeline RAG se produce un incremento adicional significativo (+10.99 puntos), ya que el modelo no tiene que leer todo el texto. La etapa de preprocesamiento para tablas añade una mejora moderada (+3,04 puntos sobre RAG básico) en este caso puesto que no hay tantas tablas, o al menos las preguntas no necesitan que se acceda a ellas. La integración de un agente culmina con más ganancia (+7,44 puntos sobre RAG con tablas), demostrando que su arquitectura solventa los problemas que tiene el RAG por si solo en las peticiones más complejas.

Conclusiones.

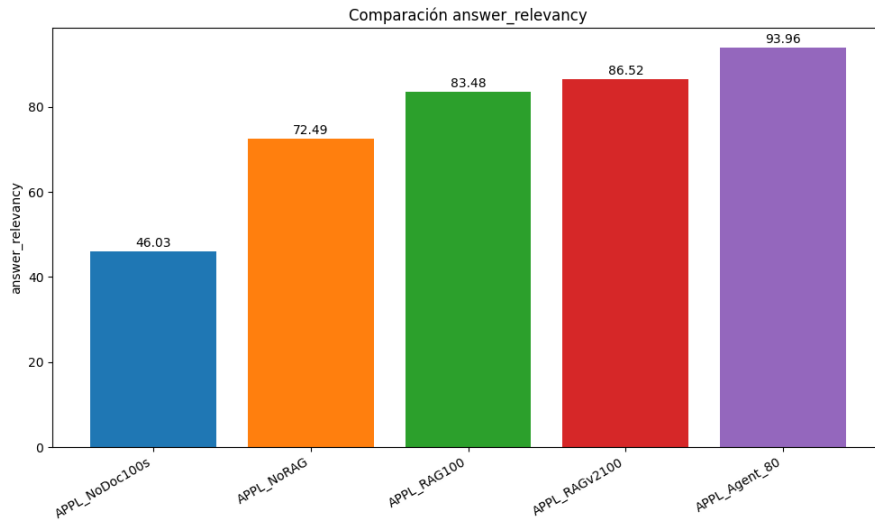


Figura 6.3: Relevancia de respuesta para cada configuración.

La combinación de recuperación de fragmentos relevantes, transformaciones especializadas para datos tabulares y control agentic resulta esencial para maximizar la relevancia de las respuestas. Cada mecanismo contribuye de forma acumulativa, confirmando que los enfoques híbridos y modulados (especialmente el agentic RAG) ofrecen el rendimiento más elevado en tareas de pregunta-respuesta basadas en conocimiento.

6.1.4. Estrategias de generación de consultas y recuperación de fragmentos

Este experimento evalúa distintas técnicas de generación de consultas sobre la base de datos y su impacto en la recuperación de chunks de contexto. Se comparan cuatro configuraciones principales (Figura 6.4):

- **Query Expansion:** Se genera una consulta optimizada a partir de la pregunta original, ampliando términos clave para mejorar la cobertura semántica y aumentar la probabilidad de recuperar fragmentos relevantes.
- **Hybrid Search:** Combina búsqueda semántica (cosine similarity) con BM25. De los dos fragmentos devueltos, uno procede del ranking BM25 y otro de la similitud de embeddings.
- **HyDE:** Basado en el método presentado en *Hypothetical Document Embeddings* [22], se sintetizan respuestas hipotéticas a partir de la consulta y,

a continuación, se emplean sus embeddings para recuperar los chunks más parecidos.²

- **RAG Fusion:** Genera múltiples consultas diferenciadas que cubren distintos aspectos del enunciado y fusiona los chunks resultantes de cada una.

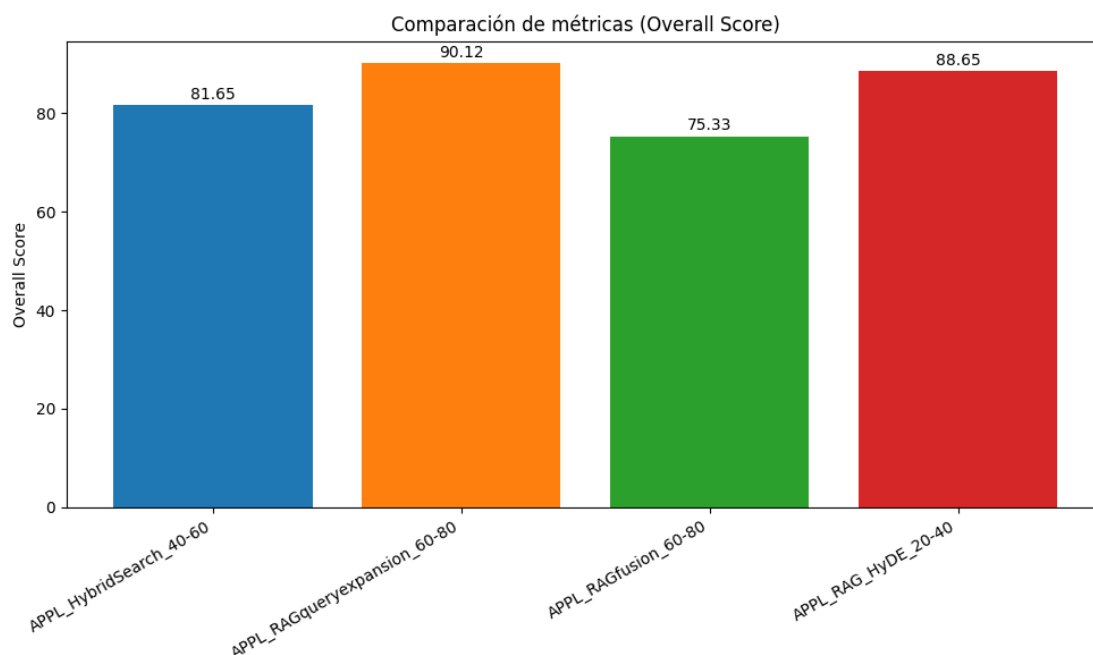


Figura 6.4: Comparación de puntuación global (Overall Score) para diferentes métodos de recuperación de contexto.

Como muestra la Figura 6.4, las configuraciones de Hybrid Search y RAG Fusion no alcanzan los mejores resultados. En el caso de Hybrid Search, la mezcla de BM25 y similitud de embeddings resulta poco efectiva: de los dos chunks retornados, uno proviene de BM25 y otro de similitud por coseno, lo que no aporta un contexto suficientemente coherente para respuestas rápidas. Por su parte, RAG Fusion introduce complejidad innecesaria para consultas sencillas y su ventaja surge únicamente en escenarios de generación de informes extensos, donde varias perspectivas de la misma pregunta pueden complementarse de forma significativa.

²HyDE (Hypothetical Document Embeddings) genera primero un “documento hipotético” a partir de la pregunta usando un modelo de lenguaje grande, y luego utiliza la representación (embedding) de ese documento generado para realizar la recuperación densa de información. De este modo, convierte un problema de baja señal en uno con contexto enriquecido, mejorando significativamente la precisión en búsquedas sin necesidad de etiquetas de relevancia.

Por otro lado, tanto Query Expansion como HyDE ofrecen mejoras sustanciales en la relevancia de las respuestas. Para validar este comportamiento se amplió el experimento a un conjunto mayor de preguntas, comparando únicamente estos dos métodos 6.5.

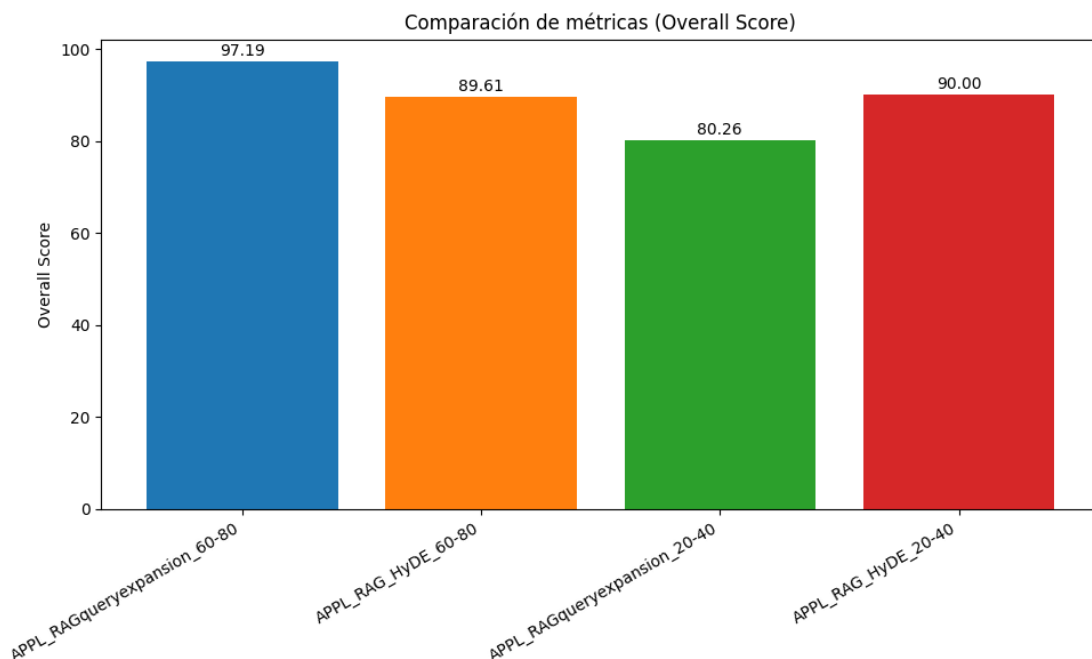


Figura 6.5: Comparación de rendimiento general entre Query Expansion y HyDE.

La Figura 6.5 revela que ambos enfoques pueden ser complementarios: en determinados casos Query Expansion supera a HyDE y, en otras consultas, HyDE obtiene un mejor desempeño. Esto sugiere que una integración de ambas técnicas podría maximizar la robustez del sistema frente a la variedad de preguntas.

6.1.5. Agente RAG

Como ya se puede apreciar de la Figura 6.4, la arquitectura agencial proporciona una mejora considerable, siendo mucho más robusta que el RAG por sí solo. Esto se justifica por sus numerosos bucles de verificación que, a pesar de añadir latencia, aumentan la fiabilidad del sistema. Para demostrar la mejora que este sistema aporta, en la Figura 6.6, se ve como mejorará en 11 puntos porcentuales aproximadamente de un sistema que tiene el query expansion al agente además de tener esta etapa inicial tiene pasos de verificación.

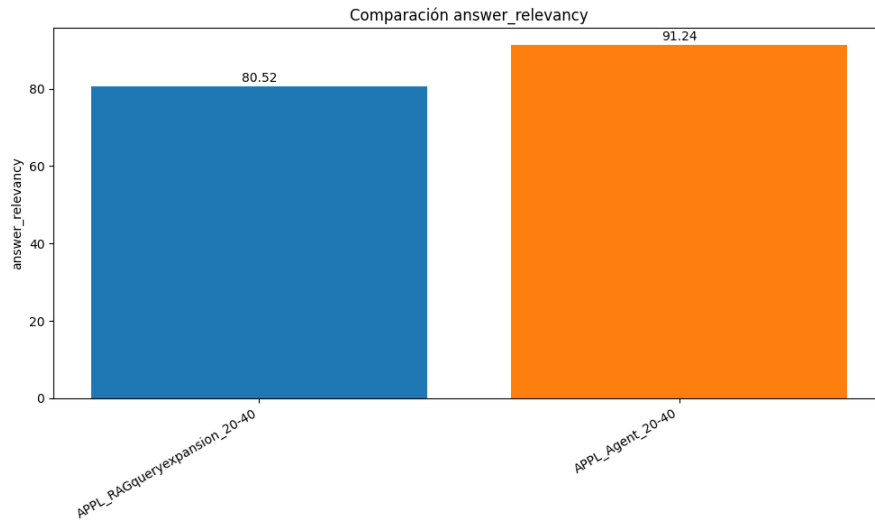


Figura 6.6: Comparativa entre agente y query expansion

6.1.6. Conclusión

En conjunto, los experimentos demuestran que un pipeline RAG bien configurado puede elevar de manera notable la calidad de las respuestas en tareas de pregunta-respuesta sobre documentos financieros extensos. Primero, mediante el análisis del número de fragmentos (chunks), se identificó que $K = 2$ ofrece el mejor compromiso entre precisión de contexto, relevancia de la respuesta y eficiencia operativa.

A continuación, la comparación entre configuraciones reveló que incorporar pasos de preprocesamiento (especialmente para tablas) y componentes agenciales aporta mejoras acumulativas: pasar de un RAG básico a un RAG con tablas supuso un aumento moderado, mientras que la integración de un agente de planificación iterativa elevó la relevancia de la respuesta considerablemente, confirmando la eficacia de los bucles de verificación para reforzar la fiabilidad general.

Por último, en las estrategias de recuperación de contexto, tanto Query Expansion como HyDE sobresalen frente a técnicas híbridas o de fusión, pero muestran comportamientos complementarios según el tipo de consulta. Esto sugiere que un enfoque combinado, que aproveche la expansión de términos y la generación de documentos hipotéticos, podría maximizar la robustez del sistema ante la diversidad de preguntas. En definitiva, los resultados avalan un diseño agencial como la solución más sólida para responder de forma precisa y eficiente en entornos de documentos financieros largos.

Capítulo 7

Discusión

7.1. Autonomía de los agentes

La autonomía de los agentes constituye un eje central de este trabajo y su aplicación efectiva depende en gran medida del contexto empresarial. En la situación actual, los agentes operan con un nivel de autonomía cuidadosamente delimitado, donde la trazabilidad de cada paso resulta ser muy importante para garantizar transparencia y confianza. Esta supervisión ofrece a las compañías un control preciso de las acciones de la inteligencia artificial y posibilita auditar la lógica subyacente a cada decisión. Sin embargo, es previsible que, a medida que la tecnología madure y las organizaciones la integren de forma natural en sus flujos de trabajo, la confianza en estos sistemas se incremente y se deleguen gradualmente más tareas sin intervención humana directa. Este avance plantea un equilibrio constante entre dotar al agente de nuevas herramientas y características (que elevan su capacidad de actuación) y mantener tiempos de respuesta competitivos; a mayor autonomía y complejidad funcional, mayor puede ser la latencia de ejecución. No obstante, por el feedback de los analistas hay múltiples procesos en los cuales priman la fiabilidad y la profundidad de los agentes, por lo que se valora aumentar la complejidad y empeorar el aspecto de la latencia a cambio de respuestas e informes de mayor calidad.

7.2. El lugar de la IA en análisis de empresas

En el ámbito del análisis financiero, la función de la inteligencia artificial y de los agentes se perfila como una herramienta de eficiencia. Las tareas sistemáticas y repetitivas, tradicionalmente realizadas de manera manual, se automatizan con mayor fiabilidad y rapidez mediante estos sistemas. Ello libera a los analistas para concentrarse en labores de mayor valor añadido, como la interpretación cualita-

tiva de resultados, la evaluación estratégica y la comunicación de hallazgos a los decisores. La consecuencia directa es una productividad notablemente superior, ya que los profesionales disponen de más tiempo para el pensamiento crítico y la creatividad. Al reducirse la carga de procesos mecánicos (naturalmente menos atractivos), el entorno laboral se vuelve más estimulante, lo que puede mejorar tanto la motivación como la retención de talento y en consecuencia, un aumento de la productividad otra vez más.

7.3. Ética y trazabilidad

Por último, la dimensión ética y la trazabilidad son fundamentales, especialmente en la funcionalidad de búsqueda en Internet. Aunque los agentes empleen algoritmos avanzados para identificar y sintetizar información, su fiabilidad depende inevitablemente de la calidad de las fuentes consultadas. Si el contenido original es inexacto, incompleto o sesgado, la respuesta generada heredará esos defectos, con el riesgo de inducir a conclusiones erróneas. Por esta razón, resulta indispensable mantener registros detallados de las fuentes empleadas, aplicar criterios estrictos de validación y promover la revisión humana en los casos más sensibles. La combinación de una IA potente con un marco ético sólido y mecanismos de auditoría constituye la base para una adopción responsable que fortalezca, en lugar de comprometer, la confianza en los sistemas de análisis automatizado.

Capítulo 8

Conclusión

8.1. Objetivos cumplidos

Se ha conseguido desarrollar unos agentes con una autonomía controlada que reducen tiempo en los procesos de análisis para los analistas. Tanto la funcionalidad RAG para trabajar con documentos en local como funcionalidad de búsqueda externa para complementar con funcionalidad en internet evitan análisis y búsquedas repetitivas. Además, el agente global proporciona la agrupación de estas dos funcionalidades y reduce la fricción de su uso con características como la memoria conversacional. La plataforma desarrollada permite un uso sencillo, permitiendo al usuario cargar archivos en el momento, especificar esquemas de respuestas y realizar peticiones con lenguaje natural, delegando análisis o trabajo en los agentes IA.

Se demuestra entonces como el IA, el RAG y los agentes son de gran utilidad para el análisis empresarial. Como se ve en la Figure 8.1, con la plataforma de

Funcionalidad	RavenPack	LSEG MRN	AlphaSense	FinChat.io	Nuestra Plataforma
Análisis estructurado de noticias	☑	☑			☑
Búsqueda semántica de documentos			☑		☑
Chat conversacional / IA generativa			☑	☑	☑
Resumen automático de documentos			☑	☑	☑
Extracción de KPIs y métricas financieras			☑	☑	☑
Generación de gráficos/tablas en respuestas				☑	☑
Agentes IA autónomos	☐	☐	☐	☐	☑

Figura 8.1: Comparativa de la herramienta desarrollada con el mercado

colaboración multi-agente, se cubren todas las funcionalidades del mercado.

8.2. Necesidad real

En la práctica profesional, los analistas se enfrentan a un volumen enorme de documentos extensos que abarcan resultados financieros, contabilidad, transcripciones y reportes de evolución de la empresa. La lectura exhaustiva de estos materiales y la extracción de información crítica demandan un esfuerzo considerable y, en ocasiones, ralentizan la toma de decisiones. La incorporación de inteligencia artificial se revela, por tanto, como una solución necesaria para gestionar la variabilidad y la magnitud de los datos disponibles. Gracias a los agentes desarrollados, las tareas de lectura repetitiva y de detección de patrones o anomalías se automatizan, lo que libera tiempo para el razonamiento estratégico del analista y favorece una comprensión global más rápida y precisa de la situación de la compañía evaluada.

8.3. Siguiendo pasos: producción

Dado el valor práctico demostrado por la herramienta, el siguiente hito consiste en su despliegue en un entorno de producción accesible a usuarios reales. Este paso permitirá que el grupo de early adopters pruebe los agentes en escenarios cotidianos y aporte retroalimentación directa sobre rendimiento, fiabilidad y usabilidad. A partir de esas observaciones se ajustarán los modelos, se optimizará la infraestructura y se mejorará la experiencia de usuario, introduciendo métricas de calidad de servicio y reforzando aspectos como la seguridad de los datos y la escalabilidad. El objetivo final es consolidar una versión estable que ofrezca resultados consistentes, facilite la integración con flujos de trabajo empresariales y sienta las bases para futuras extensiones funcionales.

8.4. Proyecto de ingeniería

En conclusión, este Trabajo de Fin de Grado ha englobado muchas de las funciones que realiza un ingeniero. La investigación ha sido una etapa clave para poder implementar metodologías modernas como el OPM y técnicas del estado del arte en el sector de la IA. El diseño ha sido fundamental como etapa previa a la creación de los agentes e incorporación de estos métodos. La presentación a los analistas y su feedback ha permitido trabajar con retroalimentación del mercado, enfocando el servicio de una manera más profesional. Y por último, y más

importante, la colaboración con los tutores ha sido de lo más beneficiosos para el proyecto, siguiendo sus pautas este proyecto ha salido hacia delante exitosamente.

Capítulo 9

Líneas futuras

Este proyecto da pie a continuar el desarrollo de nuevos en muchos aspectos. A continuación se discute como se podría proseguir y mejorar lo desarrollado. A pesar de mencionar varios caminos por donde seguir, muchos otros no mencionados también pueden tener lugar.

9.1. Optimización avanzada de los agentes

El perfeccionamiento de la arquitectura de los agentes constituye una línea prioritaria de este proyecto. Sería preciso incorporar fuentes oficiales de información financiera, tales como los repositorios de la Comisión Nacional del Mercado de Valores (CNMV) y otros organismos reguladores en sus respectivos países, cuyas interfaces de acceso suelen presentar restricciones técnicas y jurídicas. A su vez, sería de gran utilidad habilitar conectores con proveedores profesionales de datos, por ejemplo, eInforma, FactSet o plataformas similares, lo que exige una gestión centralizada y segura de credenciales junto con mecanismos de contingencia que garanticen la continuidad del servicio cuando alguna API sea inaccesible. Finalmente, la adopción de *guardrails*¹ permitirá validar cada respuesta del sistema antes de su entrega al analista, asegurando el cumplimiento normativo y la ausencia de sesgos. Aún siendo esto algo aparentemente crítico, lo es especialmente para usuario final cuando el producto está abierto a cualquiera. Sin embargo, como siempre se trata de los analistas los usuarios que tratan esta plataforma no es tan crucial dicha implementación pero debería realizarse en los próximos pasos.

¹Los guardrails o guardarrailes buscan validar tanto las entradas como las salidas de texto de los LLMs para evitar alucinaciones, omitir una posible filtración de información personal (PII), evitar usos irresponsables del agente o evitar RAG injection.

9.2. Despliegue y estrategia *DevOps*

Para dotar al proyecto de la escalabilidad necesaria, se debería valorar la migración del núcleo de servicio a un servidor ASGI basado en FastAPI, lo que facilitará el procesamiento asíncrono y la transmisión de resultados en tiempo real. Complementariamente, la adopción de una arquitectura *serverless* sobre Amazon Web Services permitiría asignar recursos de forma dinámica, reduciendo costes operativos. Esta novedosa forma de trabajar en la nube permite que los costes del despliegue vayan de la mano de su uso de forma íntegra.

9.3. Evolución de los modelos de inteligencia artificial

La plataforma se ha realizado como capa superior a los modelos IA. Por ello, se ha permitido realizar un *plug and play* que facilite la sustitución o actualización de modelos de lenguaje a medida que surjan nuevas versiones. No solo la mejora de nuevos modelos es beneficiosa, sino que esto también permite incorporar modelos personalizados, con entrenamientos específicos al sector financiero.

9.4. Protocolos de colaboración entre agentes

Los avances recientes en el protocolo Agent-to-Agent (A2A) [47] lo convierten en un candidato idóneo para dotar a la plataforma de un mecanismo de comunicación estandarizado para comunicarse con otros agentes. El anuncio oficial de Google describe una capa de mensajería neutral al modelo que engloba descubrimiento, registro de *Agent Cards*, negociación de tareas y autenticación mutua mediante firmas JWT. La adopción del protocolo A2A en el proyecto aportaría interoperabilidad directa con agentes externos basados en LangChain, AutoGen y cualquier otro framework, trazabilidad exhaustiva gracias al intercambio de metadatos firmados, y una vía de escalado horizontal mediante buses de eventos que permiten orquestar cadenas de agentes sin acoplamiento estático.

9.5. Estándares de integración de herramientas MCP

Al igual que el A2A, el Model Context Protocol (MCP) ofrece una estandarización orientada a exponer herramientas y fuentes de datos como servicios autodescriptivos que los modelos de lenguaje pueden invocar de forma segura. La publicación de Anthropic [2] presenta el protocolo como una capa extra para la

IA, donde cada servidor MCP publica un manifiesto JSON con los endpoints disponibles y sus esquemas de entrada-salida. Un estudio sistematiza su arquitectura cliente-servidor y demuestra, mediante experimentos con conjuntos de datos de GitHub y Jira, reducciones de hasta un 37% en la latencia media de invocación frente a integraciones ad-hoc [46]. Incorporar MCP en la plataforma permitiría generar adaptadores casi automáticos para nuevos servicios y habilitar la llamada de funciones complejas sin reescribir lógica de orquestación, como ocurre con todos los agentes desarrollados hasta el momento.

9.6. Memoria a largo plazo

Como una funcionalidad extra de gran utilidad para el uso de esta herramienta a diario, se valora la posibilidad de incluir una memoria a largo plazo. Como se detalla en la sección 5.3, la memoria conversacional es de gran importancia para reducir la fricción de uso de estos agentes y proporciona muchas ventajas con su implementación. Entonces, para continuar mejorando esta experiencia de usuario, la memoria a largo plazo de los agentes aportaría un valor añadido. Con esta memoria, los agentes podrían conocer preferencias de los analistas basadas en interacciones anteriores. Sería una forma de continua mejora del agente.

Puesto que enviar todos los mensajes de todas las conversaciones anteriores no es algo viable, habría que diseñar un sistema para incorporar una memoria reducida que resuma las características principales. Un método que se podría incorporar sin mucha complicación sería la de utilizar otro LLM. Este modelo resumiría conversaciones, extrayendo aspectos clave para añadir a la memoria de largo plazo. Con ello, el agente tendría los últimos mensajes (limitados) e información de conversaciones pasadas con conocimiento de las preferencias del usuario. Esto, a su vez, proporcionaría una optimización extra, reduciendo la necesidad de que los usuarios reiteren sus preferencias en todas las conversaciones nuevas.

9.7. Experiencia de usuario y evaluación de la calidad

El éxito del sistema depende, en última instancia, de la utilidad percibida por los analistas financieros. Entonces, como siguiente mejora se propone la construcción de un panel de control que presente la trazabilidad de cada respuesta, las fuentes consultadas y la estimación de confianza. Por otro lado, se establecerán indicadores clave de rendimiento tales como el tiempo medio de respuesta, la proporción de consultas atendidas y la métrica de citación de fuentes. Un mecanismo de retroalimentación explícita (basado en valoraciones positivas o negativas) que

alimentará ciclos de aprendizaje activo para que permitan: mejorar gradualmente la precisión y relevancia de los resultados.

Apéndice A

Alineación de los Objetivos de desarrollo Sostenible (ODS)

Este Trabajo de Fin de Grado, centrado en el diseño e implementación de agentes de inteligencia artificial para el análisis empresarial, se ha concebido desde el principio con el foco en los Objetivos de Desarrollo Sostenible. A lo largo de todo el proceso de investigación y desarrollo se ha procurado que cada decisión técnica, metodológica y ética repercuta positivamente en tres Objetivos de Desarrollo Sostenible: el ODS 9 (Industria, Innovación e Infraestructura), el ODS 8 (Trabajo Decente y Crecimiento Económico) y el ODS 16 (Paz, Justicia e Instituciones Sólidas).

En lo que respecta al ODS 9, el proyecto representa un ejemplo práctico de modernización industrial basada en tecnologías avanzadas de IA. La integración de LLM en la extracción y síntesis de información financiera demuestra cómo la investigación universitaria puede transferirse con agilidad al tejido productivo. Además, la arquitectura del agente facilita la interoperabilidad entre bases de datos públicas y privadas, de modo que se crea un entorno propicio para la innovación colaborativa. Por último, toda la documentación, el código comentado, lo que garantiza y fomenta la mejora continua por parte de la comunidad académica y profesional. En conjunto, estas acciones impulsan una industrialización inteligente y sostenible que reduce barreras de entrada y difunde el conocimiento.

En relación con el ODS 8, el agente de IA está pensado para complementar (y no sustituir) las competencias humanas. Al automatizar tareas repetitivas como la limpieza de datos, extracción de información y análisis evolutivo, libera tiempo del analista para actividades de interpretación estratégica y de alto valor añadido. Esta redistribución del esfuerzo incrementa la productividad y la competitividad de las empresas sin erosionar la calidad del empleo. Asimismo, al ofrecer capacidades analíticas avanzadas de forma abierta, el proyecto contribuye a democratizar la información crítica para su expansión y, con ello, a un crecimiento económico más

inclusivo y sostenible.

Por último, el ODS 16 encuentra en este trabajo un aliado para reforzar la transparencia y la rendición de cuentas en el ámbito corporativo. El agente mantiene un registro detallado de sus fuentes de datos, las transformaciones aplicadas y las métricas de evaluación, de forma que cualquier informe generado resulta fácilmente auditable tanto dentro como fuera de la organización. Al proporcionar análisis más precisos y comprensibles, se reducen las asimetrías de información entre los distintos grupos de interés, inversores, reguladores y se refuerza la confianza en los procesos de gobierno corporativo. Además, la capacidad del sistema para identificar patrones atípicos y detectar inconsistencias contables actúa como salvaguarda frente a posibles fraudes, alineándose con la meta de promover instituciones eficaces, responsables y transparentes.

En definitiva, la alineación con los ODS no es en este proyecto un mero ejercicio declarativo: se materializa en decisiones de diseño, en políticas de datos abiertos y en un marco ético que atraviesa todo el ciclo de vida del agente. La iniciativa demuestra, en definitiva, que la transformación digital puede y debe avanzar de la mano de la sostenibilidad, el empleo digno y la fortaleza institucional, integrando innovación tecnológica (ODS 9), crecimiento inclusivo (ODS 8) y transparencia empresarial (ODS 16) en un objetivo común de progreso responsable.

Apéndice B

Código Proyecto

Todo el código desarrollado y explicado durante este Trabajo de Fin de Grado se puede encontrar en los siguientes repositorios:

- RAG con filtrado de metados explicado en la sección 4.1.
- Agente RAG explicado en la sección 4.3.
- Agente de Búsqueda 1.0 explicado en la sección 5.1.
- Agente de Búsqueda 2.0 explicado en la sección 5.2.
- El Agente Global explicado en la sección 5.3.
- Evaluación de Resultados explicado en la sección 6.

En todos los repositorios se presenta un README.md para presentar la iteración de desarrollo y la forma de configurar el entorno para obtener los mismos resultados. Además, estos repositorios son accesibles navegando al perfil de Github:<https://github.com/Albertogf4/> bajo el proyecto de "@Albertogf4 Senior Project in Collaboration with Azvalor Asset Management".

Apéndice C

Vídeos Resultados

Como parte de los resultados de las distintas iteraciones, se añaden a continuación los vínculos a vídeos demostrativos:

- Agente RAG, comentado en la sección 4.3
- Agente de Búsqueda Web, comentado en la sección 5.2
- Agente Global (plataforma final), comentado en la sección 5.3.5

Apéndice D

Diagramas OPM y OPL

En este anexo se pueden encontrar todos los digramas de los agentes con sus respectivos OPL. Estos OPL son automáticamente generados por la herramienta al diseñar el diagrama, como fue explicado en la sección de metodología 1.4.

D.1. Agente RAG

OPL de la Figura D.1

1. **User** is a physical and systemic object.
2. **Query** is an informatical and systemic object.
3. **RAG Agent Processing** consists of Query Expanding, RAG Search, Reflecting, and Report Compiling.
4. **LLM** is an informatical and systemic object.
5. **Cloud Compute** is an informatical and environmental object.
6. **Internet** is an informatical and environmental object.
7. **Vector Database** is an informatical and systemic object.
8. **Vector Database** can be defined or empty.
9. **Response** is an informatical and systemic object.
10. **Query Writing** is an informatical and systemic process.
11. **Query Writing** changes Vector Database from empty to defined.

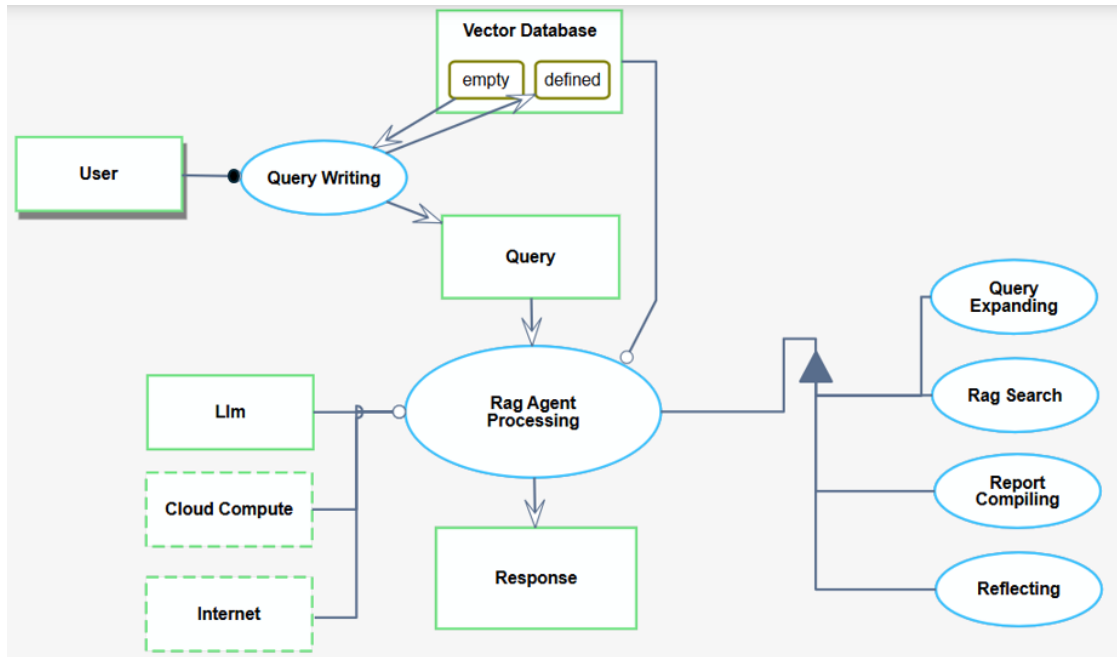


Figura D.1: OPM del Agente RAG

12. **User** handles Query Writing.
13. **Query Writing** yields Query.
14. **Query Expanding** is an informatical and systemic process.
15. **RAG Search** is an informatical and systemic process.
16. **Report Compiling** is an informatical and systemic process.
17. **Reflecting** is an informatical and systemic process.
18. **RAG Agent Processing** is an informatical and systemic process.
19. **RAG Agent Processing** requires Cloud Compute, Internet, LLM, and Vector Database.
20. **RAG Agent Processing** consumes Query.
21. **RAG Agent Processing** yields Response.

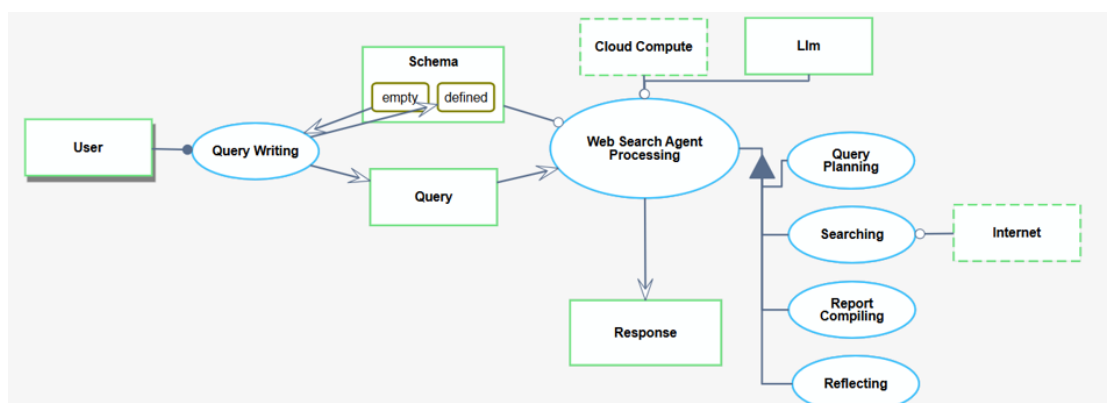


Figura D.2: OPM Agente de Búsqueda Web

D.2. Agente de Búsqueda Web

OPL de la Figura D.2

1. **User** is a physical and systemic object.
2. **Query** is an informatical and systemic object.
3. **Schema** is an informatical and systemic object.
4. **Schema** can be defined or empty.
5. **Response** is an informatical and systemic object.
6. **LLM** is an informatical and systemic object.
7. **Cloud Compute** is an informatical and environmental object.
8. **Internet** is an informatical and environmental object.
9. **Web Search Agent Processing** consists of Query Planning, Reflecting, Report Compiling, and Searching.
10. **Query Writing** is an informatical and systemic process.
11. **Query Writing** changes Schema from empty to defined.
12. **User** handles Query Writing.
13. **Query Writing** yields Query.
14. **Web Search Agent Processing** is an informatical and systemic process.

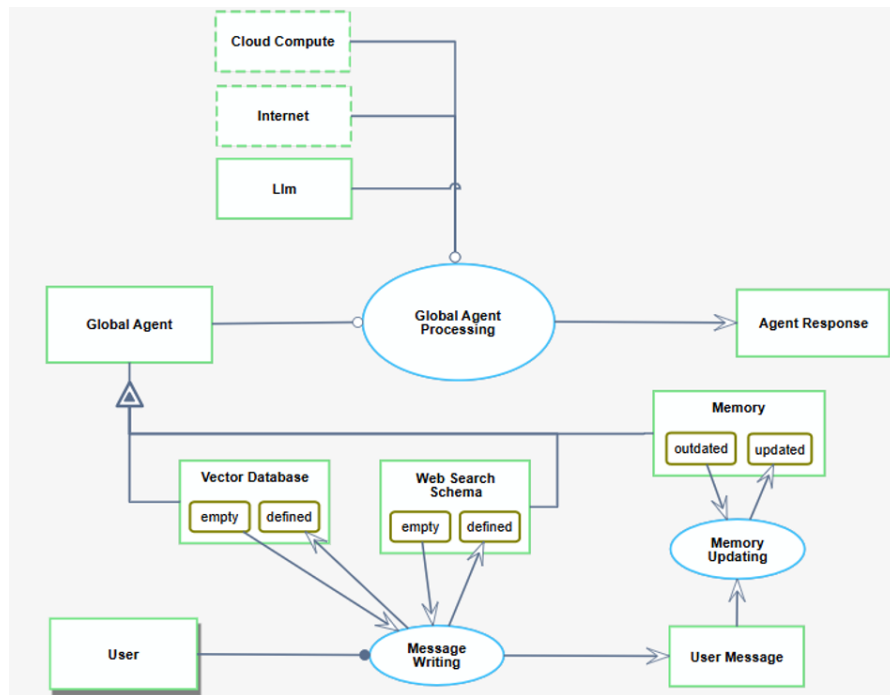


Figura D.3: Diagrama OPM Agente Global

15. **Web Search Agent Processing** requires Cloud Compute, LLM, and Schema.
16. **Web Search Agent Processing** consumes Query.
17. **Web Search Agent Processing** yields Response.
18. **Query Planning** is an informatival and systemic process.
19. **Searching** is an informatival and systemic process.
20. **Searching** requires Internet.
21. **Report Compiling** is an informatival and systemic process.
22. **Reflecting** is an informatival and systemic process.

D.3. Agente Global

OPL de la FiguraD.3, primer nivel

1. **User** is a physical and systemic object.

2. **User Message** is an informatical and systemic object.
3. **Memory of Global Agent** is an informatical and systemic object.
4. **Memory of Global Agent** can be outdated or updated.
5. **Vector Database of Global Agent** is an informatical and systemic object.
6. **Vector Database of Global Agent** can be defined or empty.
7. **Web Search Schema of Global Agent** is an informatical and systemic object.
8. **Web Search Schema of Global Agent** can be defined or empty.
9. **Global Agent** is an informatical and systemic object.
10. **Agent Response** is an informatical and systemic object.
11. **Global Agent** exhibits Memory, Vector Database, and Web Search Schema.
12. **LLM** is an informatical and systemic object.
13. **Internet** is an informatical and environmental object.
14. **Cloud Compute** is an informatical and environmental object.
15. **Message Writing** is an informatical and systemic process.
16. **Message Writing** changes Vector Database of Global Agent from empty to defined.
17. **Message Writing** changes Web Search Schema of Global Agent from empty to defined.
18. **User** handles Message Writing.
19. **Message Writing** yields User Message.
20. **Global Agent Processing** is an informatical and systemic process.
21. **Global Agent Processing** requires Global Agent.
22. **Global Agent Processing** yields Agent Response.
23. **Memory Updating** is an informatical and systemic process.

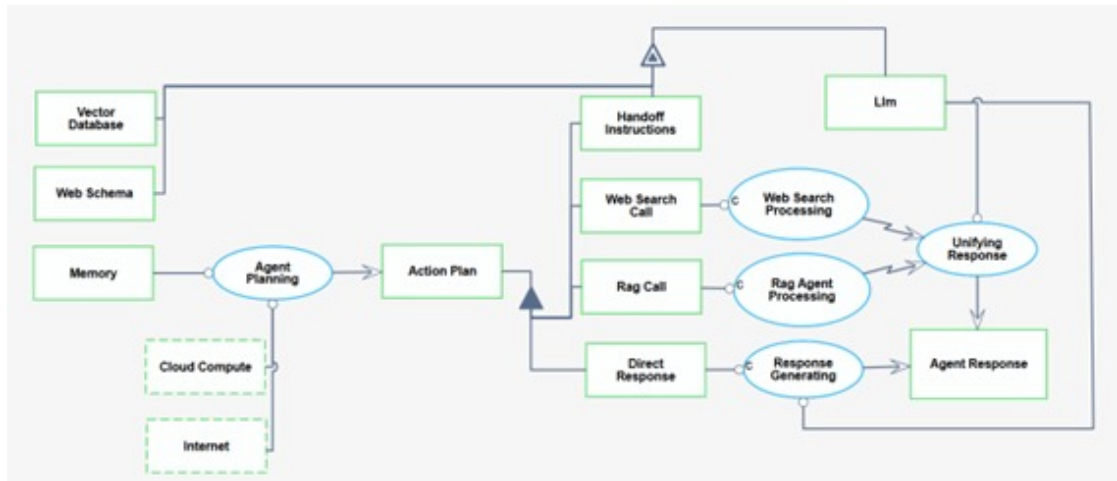


Figura D.4: Diagrama OPM Agente Global Processing

24. **Memory Updating** changes Memory of Global Agent from outdated to updated.
25. **Memory Updating** consumes User Message.
26. **Global Agent Processing** requires exactly one of Cloud Compute, Internet, and LLM.

OPL de la Figura D.4, segundo nivel

1. **Vector Database of LLM** is an informatical and systemic object.
2. **Web Schema of LLM** is an informatical and systemic object.
3. **Memory** is an informatical and systemic object.
4. **Action Plan** is an informatical and systemic object.
5. **Handoff Instructions of LLM** is an informatical and systemic object.
6. **Web Search Call** is an informatical and systemic object.
7. **RAG Call** is an informatical and systemic object.
8. **Direct Response** is an informatical and systemic object.
9. **LLM** is an informatical and systemic object.
10. **Agent Response** is an informatical and systemic object.

11. **Cloud Compute** is an informatical and environmental object.
12. **Internet** is an informatical and environmental object.
13. **Action Plan** consists of Direct Response, Handoff Instructions, RAG Call, and Web Search Call.
14. **LLM** exhibits Handoff Instructions, Vector Database, and Web Schema.
15. **Agent Planning** is an informatical and systemic process.
16. **Agent Planning** requires Cloud Compute, Internet, and Memory.
17. **Agent Planning** yields Action Plan.
18. **RAG Agent Processing** is an informatical and systemic process.
19. **RAG Agent Processing** occurs if RAG Call exists, otherwise RAG Agent Processing is skipped.
20. **RAG Agent Processing** invokes Unifying Response.
21. **Web Search Processing** is an informatical and systemic process.
22. **Web Search Processing** occurs if Web Search Call exists, otherwise Web Search Processing is skipped.
23. **Web Search Processing** invokes Unifying Response.
24. **Response Generating** is an informatical and systemic process.
25. **Response Generating** requires LLM.
26. **Response Generating** occurs if Direct Response exists, otherwise Response Generating is skipped.
27. **Response Generating** yields Agent Response.
28. **Unifying Response** is an informatical and systemic process.
29. **Unifying Response** requires LLM.
30. **Unifying Response** yields Agent Response.

Bibliografía

- [1] AlphaSense Inc. *AlphaSense: Plataforma de inteligencia de mercado*. Sitio web oficial. Accedido el 4 de junio de 2025. 2025. URL: <https://www.alpha-sense.com/>.
- [2] Anthropic. *Introducing the Model Context Protocol*. Anthropic News. Nov. de 2024. URL: <https://www.anthropic.com/news/model-context-protocol> (visitado 18-06-2025).
- [3] Aki Ariga y contributors. *tabula-py: Python wrapper of tabula-java*. 2024. URL: <https://pypi.org/project/tabula-py/>.
- [4] Akari Asai et al. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. 2023. arXiv: 2310.11511 [cs.CL]. URL: <https://arxiv.org/abs/2310.11511>.
- [5] Azvalor Asset Management SGIIC SAU. *Azvalor*. Accedido el 2025-06-05. 2025. URL: <https://www.azvalor.com/>.
- [6] Bloomberg L.P. *Bloomberg*. Accessed: 2025-06-04. 2025. URL: <https://www.bloomberg.com>.
- [7] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: <https://arxiv.org/abs/2005.14165>.
- [8] Sondos Mahmoud Bsharat, Aidar Myrzakhan y Zhiqiang Shen. *Principled Instructions Are All You Need for Questioning LLaMA-1/2, GPT-3.5/4*. 2024. arXiv: 2312.16171 [cs.CL]. URL: <https://arxiv.org/abs/2312.16171>.
- [9] Camelot Development Team. *Camelot: PDF Table Extraction for Humans*. 2024. URL: <https://camelot-py.readthedocs.io/>.
- [10] Harrison Chase. *LangChain*. <https://github.com/langchain-ai/langchain>. Accessed: 2025-06-07. 2023.

- [11] Jian Chen et al. “FinTextQA: A Dataset for Long-form Financial Question Answering”. En: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2024, págs. 6025-6047. DOI: 10.18653/v1/2024.acl-long.328. URL: <http://dx.doi.org/10.18653/v1/2024.acl-long.328>.
- [12] Zhiyu Chen et al. *FinQA: A Dataset of Numerical Reasoning over Financial Data*. 2022. arXiv: 2109.00122 [cs.CL]. URL: <https://arxiv.org/abs/2109.00122>.
- [13] ChromaDB Contributors. *ChromaDB: Open-Source Embeddings Database*. <https://github.com/chroma-core/chroma>. Accessed: 31 Jan. 2025. 2025.
- [14] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: <https://arxiv.org/abs/1810.04805>.
- [15] Dov Dori. *Model-Based Systems Engineering with OPM and SysML*. Springer, 2016. URL: <https://www.springerprofessional.de/en/model-based-systems-engineering-with-opm-and-sysml/10245912>.
- [16] Enterprise Systems Modeling Laboratory. *OPCloud: Herramienta de modelado basada en OPM*. Accedido el 2025-06-05. 2025. URL: <https://www.opcloud.tech>.
- [17] Equipo de Tavily. *Tavily: software de búsqueda*. <https://tavily.com/>. Accessed 14 Feb. 2025. 2025.
- [18] ExplodingGradients. *Ragas: Supercharge Your LLM Application Evaluations*. <https://github.com/explodinggradients/ragas>. 2024.
- [19] FactSet Research Systems Inc. *FactSet*. Accessed: 2025-06-04. 2025. URL: <https://www.factset.com/>.
- [20] Mathieu Fenniak et al. *The PyPDF2 library*. See CONTRIBUTORS at PyPDF2 documentation. 2022. URL: <https://pypi.org/project/PyPDF2/>.
- [21] FinChat.io. *FinChat.io: Plataforma de investigación financiera impulsada por inteligencia artificial*. Sitio web oficial. Accedido el 4 de junio de 2025. 2025. URL: <https://finchat.io/>.
- [22] Luyu Gao et al. *Precise Zero-Shot Dense Retrieval without Relevance Labels*. 2022. arXiv: 2212.10496 [cs.IR]. URL: <https://arxiv.org/abs/2212.10496>.

- [23] Sepp Hochreiter y Jürgen Schmidhuber. “Long Short-Term Memory”. En: *Neural Comput.* 9.8 (nov. de 1997), págs. 1735-1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [24] Sepp Hochreiter y Jürgen Schmidhuber. “Long Short-Term Memory”. En: *Neural Computation* 9.8 (1997), págs. 1735-1780. DOI: 10.1162/neco.1997.9.8.1735.
- [25] George Hotz y Smol Developers. *smol-ai/developer: minimal agents that self-improve*. <https://github.com/smol-ai/developer>. Accessed: 2025-06-07. 2023.
- [26] Roger G. Ibbotson y Rex A. Sinquefeld. *Stocks, Bonds, Bills, and Inflation: Historical Returns (1926–1987)*. Chicago, IL: CFA Institute Research y Policy Center, 1989.
- [27] Soyeong Jeong et al. *Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity*. 2024. arXiv: 2403.14403 [cs.CL]. URL: <https://arxiv.org/abs/2403.14403>.
- [28] Mandar Joshi et al. “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension”. En: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada: Association for Computational Linguistics, jul. de 2017.
- [29] LangChain. *LangGraph - State Machine Library for LLM Applications*. <https://github.com/langchain-ai/langgraph>. Accessed: 2025-06-07. 2024.
- [30] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. En: *Advances in Neural Information Processing Systems* 33 (2020), págs. 9459-9474.
- [31] LlamaIndex Development Team. *LlamaIndex (GPT-Index): Data framework for LLM applications*. https://github.com/run-llama/llama_index. Último acceso: June 2025. 2025.
- [32] LlamaIndex Team. *LlamaParse*. 2025. URL: https://docs.llamaindex.ai/en/stable/llama_cloud/llama_parse/.
- [33] London Stock Exchange Group. *LSEG News Analytics: Análisis de noticias políticas y generales*. Sitio web oficial. Accedido el 4 de junio de 2025. 2025. URL: <https://www.lseg.com/en/data-analytics/financial-data/financial-news-coverage/political-news-feeds-analysis/news-analytics>.

- [34] Ali Modarressi et al. *NoLiMa: Long-Context Evaluation Beyond Literal Matching*. 2025. arXiv: 2502.05167 [cs.CL]. URL: <https://arxiv.org/abs/2502.05167>.
- [35] João Moura y CrewAI Contributors. *CrewAI: Framework for Multi-Agent LLM Workflows*. <https://github.com/joaomdmoura/crewAI>. Accessed: 2025-06-07. 2024.
- [36] n8n.io. *n8n - Workflow Automation for Technical People*. <https://n8n.io>. Accessed: 2025-06-07. 2020.
- [37] New York Life Investments. *Ibbotson® S&P® Yearbook: 1926–2024*. Inf. téc. Rentabilidad geométrica anual compuesta: 11,8 % small caps, 10,4 % large caps, 5,0 % bonos gubernamentales, 3,3 % Treasury bills. New York Life Investments, 2024.
- [38] OpenAI. *Chat GPT Search*. Public release, <https://openai.com/index/introducing-chatgpt-search/>. Accessed: 14 Feb. 2025. Feb. de 2025.
- [39] Kalyani Pakhale. *Comprehensive Overview of Named Entity Recognition: Models, Domain-Specific Applications and Challenges*. 2023. arXiv: 2309.14084 [cs.CL]. URL: <https://arxiv.org/abs/2309.14084>.
- [40] RavenPack International S.L. *RavenPack: Plataforma de análisis de datos no estructurados*. Sitio web oficial. Accedido el 4 de junio de 2025. 2025. URL: <https://www.ravenpack.com/>.
- [41] Yousef Saeedian. *Financial Q&A – 10k*. <https://www.kaggle.com/datasets/yousefsaeedian/financial-q-and-a-10k/data>. Updated a year ago; Accessed: 22 June 2025. 2024.
- [42] Sander Schulhoff et al. *The Prompt Report: A Systematic Survey of Prompt Engineering Techniques*. 2025. arXiv: 2406.06608 [cs.CL]. URL: <https://arxiv.org/abs/2406.06608>.
- [43] Ken Schwaber y Jeff Sutherland. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Accedido el 2025-06-05. Nov. de 2020. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>.
- [44] Noa Shinn, Yueming Zhang, Yujia Shen et al. “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework”. En: *arXiv preprint arXiv:2309.12307* (2023). URL: <https://arxiv.org/abs/2309.12307>.
- [45] Jeremy Singer-Vine, Samkit Jain y contributors. *pdfplumber: Plumb a PDF for detailed information*. 2024. URL: <https://github.com/jsvine/pdfplumber>.

- [46] Aditi Singh et al. “A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs)”. En: *Preprints* (2025). DOI: 10.20944/preprints202504.0245.v1.
- [47] Rao Surapaneni et al. *Announcing the Agent2Agent Protocol (A2A)*. Google Developers Blog. Abr. de 2025. URL: <https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interopability/> (visitado 18-06-2025).
- [48] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: <https://arxiv.org/abs/1706.03762>.
- [49] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. En: *Advances in Neural Information Processing Systems*. Ed. por S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, págs. 24824-24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.
- [50] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: <https://arxiv.org/abs/2201.11903>.
- [51] Shunyu Yao et al. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. arXiv: 2210.03629 [cs.CL]. URL: <https://arxiv.org/abs/2210.03629>.
- [52] Shunyu Yao et al. “Tree of Thoughts: Deliberate Problem Solving with Large Language Models”. En: *Advances in Neural Information Processing Systems*. Ed. por A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, págs. 11809-11822. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

