ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

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

ClaudIA: Desarrollo de funcionalidades mediante inteligencia artificial orientadas al Contract Life Management (CLM) y a la anonimización de datos sensibles.

Autor
José Guardo Medina

Dirigido por
Luis Francisco Sánchez Merchante
Ricardo Ferrero
Mercedes de Prada

Madrid
Julio 2025

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

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

ClaudIA: Desarrollo de funcionalidades mediante inteligencia artificial orientadas al Contract Life Management (CLM) y a la anonimización de datos sensibles.

Autor
José Guardo Medina

Dirigido por
Luis Francisco Sánchez Merchante
Ricardo Ferrero
Mercedes de Prada

Madrid
Julio 2025

José Guardo Medina, declara bajo su responsabilidad, que el Proyecto con título **ClaudIA: Desarrollo de funcionalidades mediante inteligencia artificial orientadas al Contract Life Management (CLM) y a la anonimización de datos sensibles.** 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.: .. Luis Fco Sánchez 2025.07.03 18:36:21 +02'00' Fecha: ..... / ..... / .....

El Co-Director del proyecto

Ricardo Ferrero

Fdo.: ..... Ricardo Ferrero 2025.07.03 23:55:17 +02'00' Fecha: ..... / ..... / .....

La Co-Directora del proyecto

Mercedes de Prada

Fdo.: ..... Fecha: ..... / ..... / .....

Resumen

ClaudIA: Desarrollo de funcionalidades mediante inteligencia artificial orientadas al Contract Life Management (CLM) y a la anonimización de datos sensibles.
Autor: Guardo Medina, José.
Director: Sánchez Merchante, Luis Francisco.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

ClaudIA es el nombre de la infraestructura completa diseñada en este Trabajo de Fin de Grado, compuesta por herramientas LegalTech orientadas a resolver necesidades reales del ámbito jurídico, validadas con abogados especializados que han acompañado y orientado el desarrollo técnico del proyecto. La decisión de centrar el trabajo en el sector LegalTech surge de mi participación activa durante dos años en el Observatorio LegalTech de ICADE - Garrigues, experiencia que me permitió explorar en profundidad las sinergias entre el derecho y las tecnologías más punteras.

Como resultado, el proyecto se articula en torno a dos soluciones complementarias: por un lado, un sistema de anonimización de Personal Identifiable Information (PII) integrado en Microsoft Word, capaz de proteger datos sensibles de manera precisa y automatizada; y por otro, un sistema avanzado de *Contract Lifecycle Management* (CLM), diseñado específicamente para contratos EPC (Engineering, Procurement and Construction), basado en una arquitectura híbrida que combina modelos de inteligencia artificial, técnicas RAG (Retrieval-Augmented Generation) y Knowledge Graphs.

La elección de los contratos EPC como caso de uso no es casual: estos documentos integran dimensiones jurídicas, técnicas, financieras y temporales que los convierten en uno de los formatos contractuales más complejos y exigentes. Enfrentar esta complejidad desde el inicio ha permitido validar la capacidad de las soluciones propuestas para abordar escenarios de alta densidad legal, demostrando al mismo tiempo su escalabilidad y aplicabilidad a otros tipos de contratos más convencionales. En este sentido, el trabajo no solo destaca por su componente tecnológico innovador, sino también por su orientación práctica y su potencial de impacto en el ejercicio profesional del derecho.

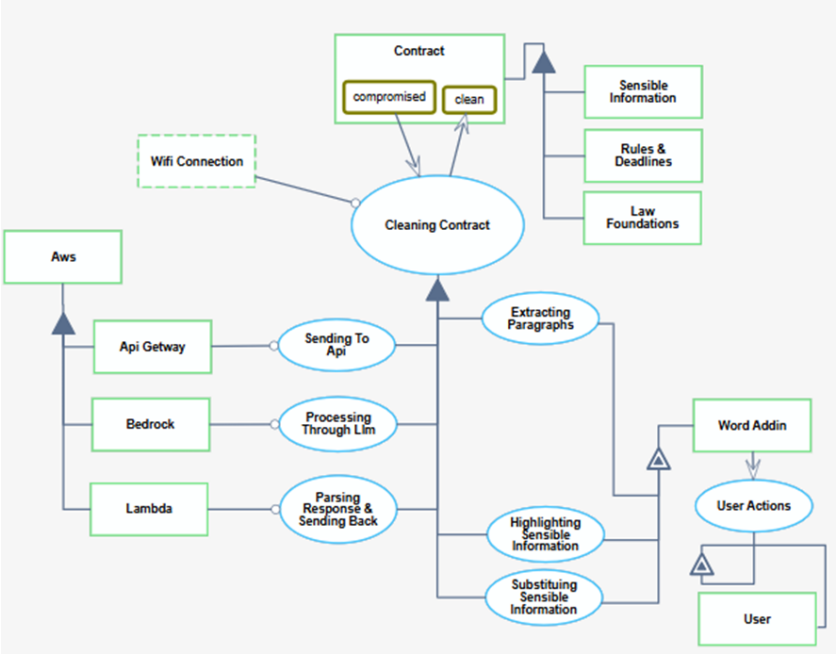


Figure 1: Diagrama OPM Sistema de Anonimización

Palabras Clave:
LLMs, RAG, IA, WORD ADDIN, CLM, Knowledge Graph, Anonimización, PII.

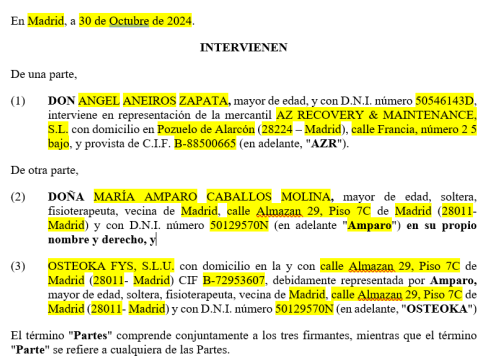
Sistema de Anonimización

Como parte del proyecto ClaudIA, se ha desarrollado un sistema de anonimización de datos sensibles5 integrado en Microsoft Word, diseñado para proteger información personal y confidencial en documentos legales. Esta herramienta permite al usuario analizar cualquier texto, identificar automáticamente datos como nombres, DNIs, direcciones o referencias económicas, y sustituirlos por etiquetas seguras.

El sistema está conectado a una arquitectura en la nube (AWS) que ejecuta modelos de inteligencia artificial especializados en privacidad, garantizando un análisis preciso y seguro sin almacenar los datos procesados. Todo el flujo, desde la detección hasta la sustitución y reversión, se realiza de forma rápida e intuitiva desde el propio Word, sin necesidad de conocimientos técnicos por parte del usuario.

Esta solución responde a una necesidad real detectada en el ámbito jurídico, especialmente en contextos donde el cumplimiento del RGPD es clave. Su integración directa en herramientas de uso cotidiano, como Word, la convierte en una propuesta práctica, escalable y de alto valor para despachos y equipos legales.

Por último, el resumen de tu TFG sirve como muestra breve de lo que los lectores pueden esperar sobre el estudio. Este facilita la comprensión y aporta



En Madrid, a 30 de Octubre de 2024.

INTERVIENEN

De una parte,

(1) **DON** ANGEL ANEIROS ZAPATA**,** mayor de edad, y con D.N.I. número 50546143D, interviene en representación de la mercantil AZ RECOVERY & MAINTENANCE, S.L. con domicilio en Pozuelo de Alarcón (28224 – Madrid), calle Francia, número 2 5 bajo, y provista de C.I.F. B-88500665 (en adelante, "**AZR**").

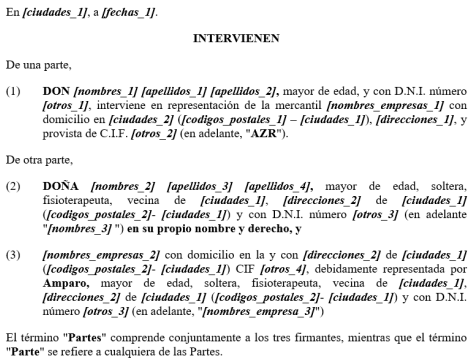
De otra parte,

(2) **DOÑA** MARÍA AMPARO CABALLOS MOLINA**,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante "**Amparo**") **en su propio nombre y derecho,** y

(3) OSTEOKA FYS, S.L.U. con domicilio en la y con calle Almazan 29, Piso 7C de Madrid (28011- Madrid) CIF B-72953607, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante, "**OSTEOKA**")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 2: PII Marcado



En ***[ciudades_1]***, a ***[fechas_1]***.

INTERVIENEN

De una parte,

(1) **DON** ***[nombres_1] [apellidos_1] [apellidos_2],*** mayor de edad, y con D.N.I. número ***[otros_1]***, interviene en representación de la mercantil ***[nombres_empresas_1]*** con domicilio en ***[ciudades_2]*** (***[codigos_postales_1]*** – ***[ciudades_1]***), ***[direcciones_1]***, y provista de C.I.F. ***[otros_2]*** (en adelante, "**AZR**").

De otra parte,

(2) **DOÑA** ***[nombres_2] [apellidos_3] [apellidos_4],*** mayor de edad, soltera, fisioterapeuta, vecina de ***[ciudades_1]***, ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) y con D.N.I. número ***[otros_3]*** (en adelante "***[nombres_3]*** ") **en su propio nombre y derecho,** y

(3) ***[nombres_empresas_2]*** con domicilio en la y con ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) CIF ***[otros_4]***, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de ***[ciudades_1]***, ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) y con D.N.I. número ***[otros_3]*** (en adelante, "***[nombres_empresa_3]***")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 3: PII Extraído

contexto a tu trabajo. Un resumen bien escrito permite que la información más compleja de tu trabajo sea mejor comprendida.

Resultados obtenidos

A continuación se muestra un ejemplo representativo de una cláusula legal antes y después de ser procesada por el sistema de anonimización.

Sistema de CLM

El segundo módulo del proyecto 8 propone una solución para organizar y comprender contratos complejos, como los EPC, mediante técnicas de inteligencia artificial y grafos de conocimiento. A partir del texto del contrato, el sistema agrupa cláusulas similares, les asigna etiquetas jurídicas y detecta relaciones entre ellas, construyendo así un mapa semántico del contenido. Esta información estructurada se utiliza luego para responder preguntas legales concretas, generando eventos clave del contrato (como fechas límite u obligaciones) y mostrando los resultados en un calendario visual interactivo. Además, cada respuesta viene acompañada de citas textuales que justifican su validez, aportando transparencia y confianza. La arquitectura completa está pensada para ser escalable, modular y útil tanto para abogados como para gestores de proyecto. Los resultados obtenidos en la aplicación del sistema a contratos EPC han sido satisfactorios. El grafo de conocimiento ha permitido representar de forma estructurada y navegable las relaciones clave entre cláusulas contractuales, facilitando su comprensión y análisis legal. A su vez, el calendario contractual ha demostrado ser una herramienta eficaz para extraer y organizar hitos temporales, derechos y obligaciones, ofreciendo una visualización clara y accionable del ciclo de vida del contrato. Ambos módulos evidencian que este enfoque, basado en modelos de lenguaje y estructuras semánticas, es capaz de gestionar contratos complejos y resulta escalable a otros tipos contractuales más simples.

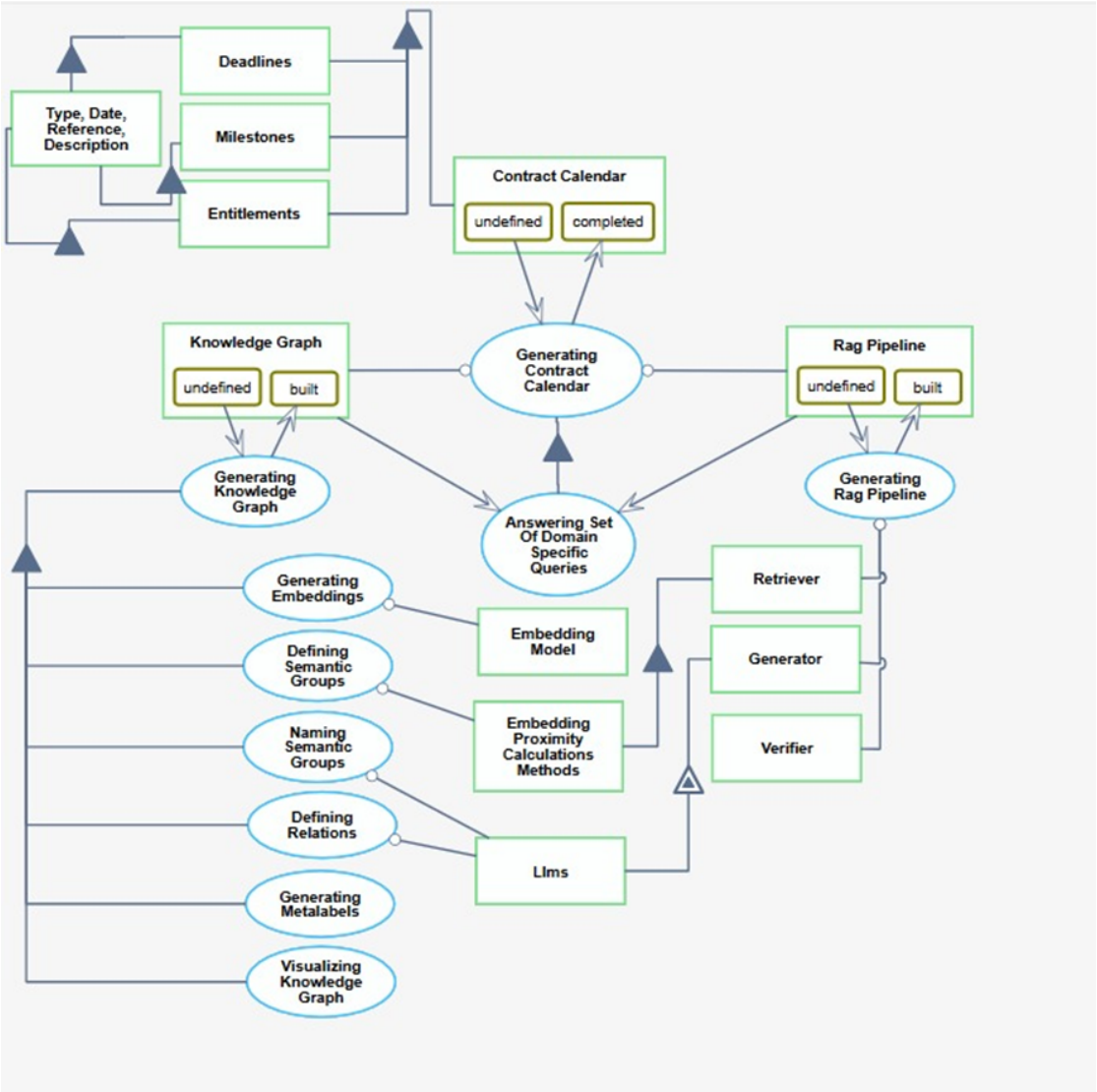


Figure 4: Diagrama OPM del CLM

Abstract

ClaudIA: Development of Artificial Intelligence Functionalities for Contract Lifecycle Management (CLM) and Sensitive Data Anonymization.
Author: Guardo Medina, José.
Supervisor: Sánchez Merchante, Luis Francisco.
Partner Institution: ICAI – Universidad Pontificia Comillas

ClaudIA is the name of the full infrastructure designed in this Bachelor's Thesis, consisting of LegalTech tools aimed at solving real needs in the legal domain. These tools have been validated by expert lawyers who have guided the technical development of the project. The decision to focus on the LegalTech sector stems from my active participation for two years in the ICADE - Garrigues LegalTech Observatory, an experience that allowed me to deeply explore the synergies between law and cutting-edge technologies.

As a result, the project is structured around two complementary solutions: on the one hand, a Personal Identifiable Information (PII) anonymization system integrated into Microsoft Word, capable of precisely and automatically protecting sensitive data; and on the other, an advanced Contract Lifecycle Management (CLM) system, specifically designed for EPC (Engineering, Procurement and Construction) contracts. This system is based on a hybrid architecture that combines artificial intelligence models, Retrieval-Augmented Generation (RAG) techniques, and Knowledge Graphs.

The choice of EPC contracts as a use case is deliberate: these documents encompass legal, technical, financial, and temporal dimensions, making them one of the most complex and demanding contract formats. Addressing this complexity from the outset enabled the validation of the proposed solutions in high-density legal scenarios, while also demonstrating their scalability and applicability to more conventional contract types. In this sense, the project stands out not only for its innovative technological approach but also for its practical orientation and potential impact on legal practice.

Keywords:
LLMs, RAG, AI, WORD ADD-IN, CLM, Knowledge Graph, Anonymization, PII.

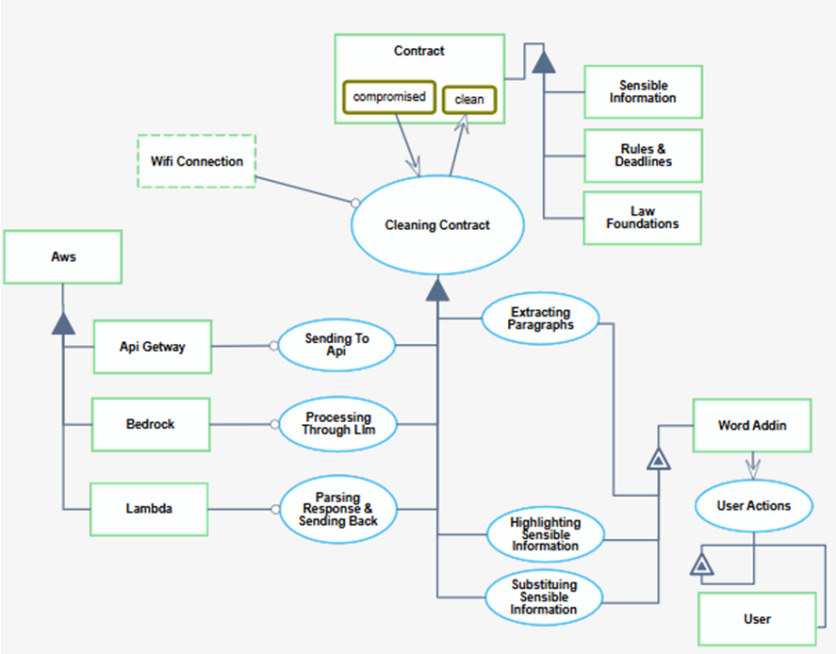


Figure 5: OPM Diagram of the Anonymization System

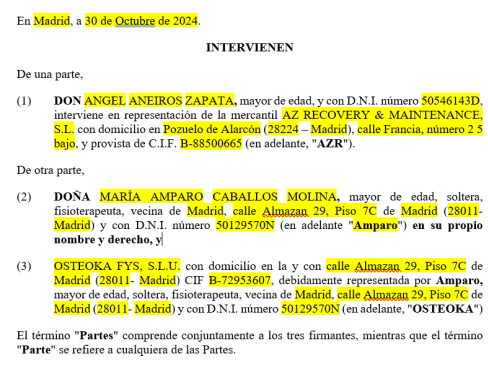
Anonymization System

As part of the ClaudIA project, a sensitive data anonymization system has been developed 5, integrated into Microsoft Word. It is designed to protect personal and confidential information in legal documents. This tool allows users to analyze any text, automatically identify data such as names, IDs, addresses, or financial references, and replace them with secure labels.

The system is connected to a cloud-based architecture (AWS) that runs artificial intelligence models specialized in privacy, ensuring accurate and secure analysis without storing the processed data. The entire process—from detection to replacement and restoration—is carried out quickly and intuitively within Word itself, without requiring technical knowledge from the user.

This solution addresses a real need identified in the legal sector, particularly in contexts where GDPR compliance is critical. Its direct integration into everyday tools like Word makes it a practical, scalable, and high-value proposal for law firms and legal teams.

Finally, the abstract serves as a brief summary of what readers can expect from the full thesis. It provides context and improves comprehension of the work. A well-written abstract helps make the more complex aspects of the project easier to understand.



En Madrid, a 30 de Octubre de 2024.

INTERVIENEN

De una parte,

(1) **DON** ANGEL ANEIROS ZAPATA**,** mayor de edad, y con D.N.I. número 50546143D, interviene en representación de la mercantil AZ RECOVERY & MAINTENANCE, S.L. con domicilio en Pozuelo de Alarcón (28224 – Madrid), calle Francia, número 2 5 bajo, y provista de C.I.F. B-88500665 (en adelante, "**AZR**").

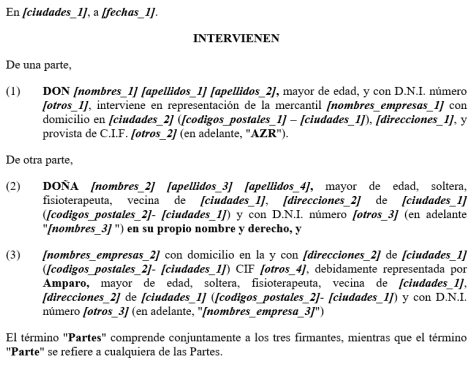
De otra parte,

(2) **DOÑA** MARÍA AMPARO CABALLOS MOLINA**,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante "**Amparo**") **en su propio nombre y derecho, y**

(3) OSTEOKA FYS, S.L.U. con domicilio en la y con calle Almazan 29, Piso 7C de Madrid (28011- Madrid) CIF B-72953607, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante, "**OSTEOKA**")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 6: Marked PII



En ***[ciudades_1]***, a ***[fechas_1]***.

INTERVIENEN

De una parte,

(1) **DON** ***[nombres_1] [apellidos_1] [apellidos_2]*****,** mayor de edad, y con D.N.I. número ***[otros_1]***, interviene en representación de la mercantil ***[nombres_empresas_1]*** con domicilio en ***[ciudades_2]*** (***[codigos_postales_1]*** – ***[ciudades_1]***), ***[direcciones_1]***, y provista de C.I.F. ***[otros_2]*** (en adelante, "**AZR**").

De otra parte,

(2) **DOÑA** ***[nombres_2] [apellidos_3] [apellidos_4]*****,** mayor de edad, soltera, fisioterapeuta, vecina de ***[ciudades_1]***, ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) y con D.N.I. número ***[otros_3]*** (en adelante "***[nombres_3]*** ") **en su propio nombre y derecho, y**

(3) ***[nombres_empresas_2]*** con domicilio en la y con ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) CIF ***[otros_4]***, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de ***[ciudades_1]***, ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) y con D.N.I. número ***[otros_3]*** (en adelante, "***[nombres_empresa_3]***")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 7: Extracted PII

Results

Below is a representative example of a legal clause before and after being processed by the anonymization system.

CLM System

The second module of the project 8 presents a solution for organizing and understanding complex contracts—such as EPC—using artificial intelligence techniques and knowledge graphs. From the contract text, the system clusters similar clauses, assigns them legal labels, and detects relationships between them, thereby building a semantic map of the content. This structured information is then used to answer specific legal questions by generating key contractual events (such as deadlines or obligations) and displaying the results in an interactive visual calendar. Each answer is accompanied by textual citations that justify its validity, providing transparency and trust.

The full architecture is designed to be scalable, modular, and useful for both legal professionals and project managers. The application of this system to EPC contracts yielded satisfactory results. The knowledge graph made it possible to represent key contractual relationships in a structured and navigable way, improving understanding and legal analysis. Meanwhile, the contractual calendar proved to be an effective tool for extracting and organizing milestones, rights, and obligations, offering a clear and actionable visualization of the contract lifecycle. Both modules demonstrate that this approach—based on language models and semantic structures—is capable of handling complex contracts and is scalable to simpler ones.

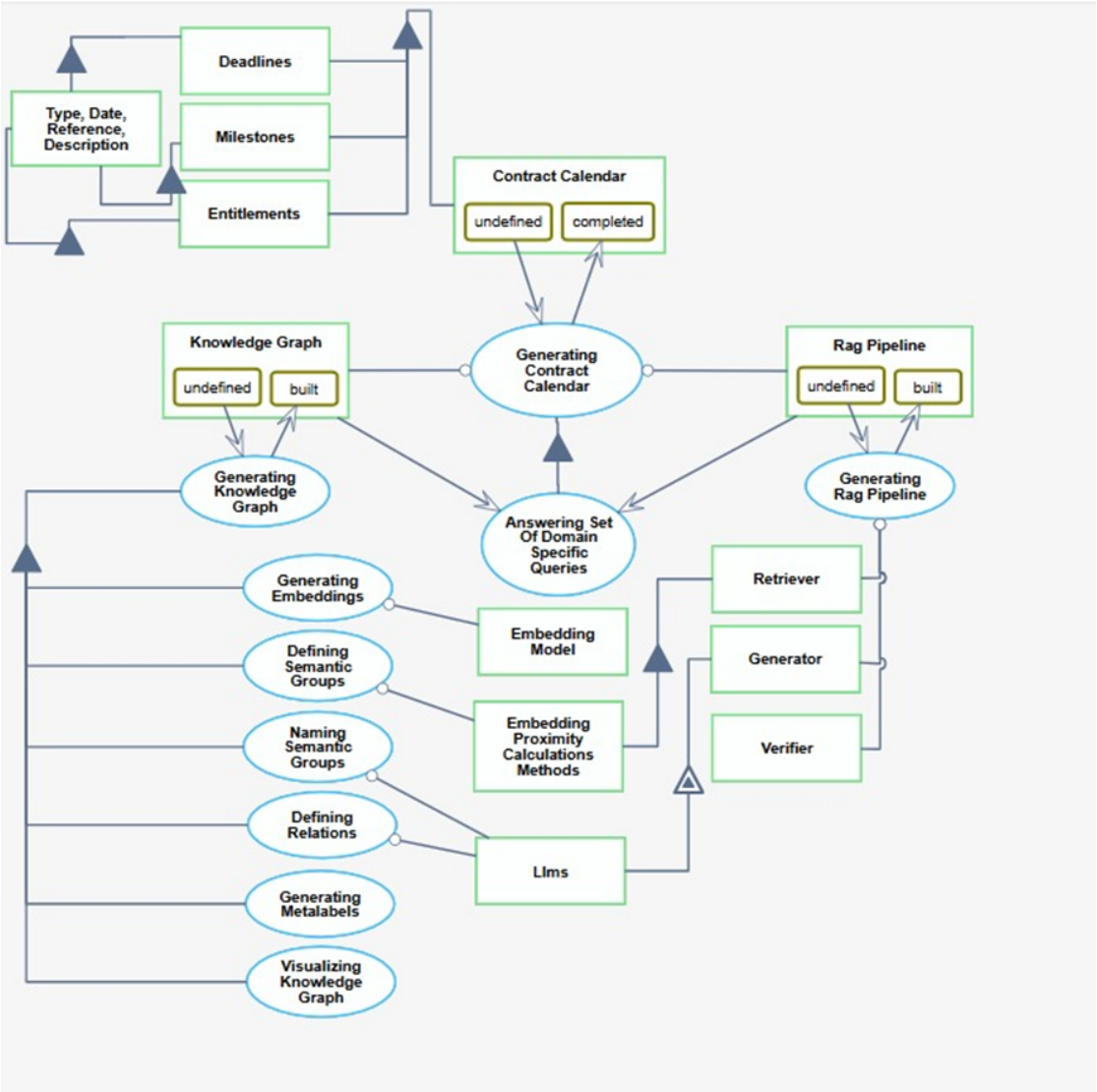


Figure 8: OPM Diagram of the CLM System

A mis padres
por su entrega y cariño.

If the computer scientist is a toolsmith,
and if our delight is to fashion power tools
and amplifiers for minds,
we must partner with those,
who will use our tools,
those whose intelligences we hope to amplify.
Fred Brooks

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mis directores de TFG, Luis y Ricardo, por su paciencia, constancia y dedicación a lo largo de todo el proceso. La visión tecnológica que me han transmitido, unida a su capacidad para ampliar el enfoque más allá del plano puramente técnico hacia una perspectiva estratégica y comercial, ha sido de un valor incalculable. Su guía no solo ha enriquecido este proyecto, sino también mi forma de entender la ingeniería como una disciplina aplicada, creativa y orientada al impacto real.

No puedo dejar de mencionar a Raquel y Alberto, cuyo apoyo ha sido fundamental en este proyecto. Gracias a ellos pude identificar una necesidad real dentro del sector legal y orientar mi trabajo hacia la construcción de una solución con impacto tangible. Su orientación constante, tanto en la definición del problema como en el enfoque práctico del desarrollo, ha sido clave para convertir este proyecto en una herramienta verdaderamente útil.

Por último, quiero agradecer a mis padres por su incansable escucha, apoyo incondicional y absoluto cariño. Sin ellos no me hubiese enfrentado a este reto con tanta determinación y confianza en mi mismo.

Contents

1 Introducción **1**
- 1.1 Contexto . . . 1
- 1.2 Motivación y Alcance . . . 2
- 1.3 Objetivos . . . 3
- 1.4 Metodología . . . 4

2 Estado del Arte **7**
- 2.1 Expansión de las Tecnologías LegalTech . . . 7
  - 2.1.1 Contract Life Management en el mercado nacional . . . 7
  - 2.1.2 Contract Life Management en el mercado internacional . . . 8
  - 2.1.3 Soluciones para la Anonimización de Datos Sensibles . . . 9
- 2.2 Estado actual de LLMs, RAG y Sistemas de Agentes . . . 11
  - 2.2.1 Competencia Reciente en LLMs . . . 11
  - 2.2.2 Integración de información específica mediante Retrieval Augmented Generation (RAG) . . . 12
  - 2.2.3 Agentes de IA y Function Calling . . . 14

3 Introducción a la arquitectura del sistema **17**
- 3.1 Estructura y Dessarollo de un AddIn de Word . . . 18
  - 3.1.1 Nucleo declarativo del AddIn: Manifest . . . 18
  - 3.1.2 Estructura de Aplicación Web . . . 19
  - 3.1.3 Archivos Estátios . . . 20
  - 3.1.4 Servidor local para el desarrollo . . . 20
  - 3.1.5 Uso de la API JavaScript de Office . . . 21
  - 3.1.6 Resumen de la estructura estándar de un AddIn de Word . . . 21

4 Sistema de Anonimizacion de Datos Sensibles **23**
- 4.1 Modelos NER . . . 23
  - 4.1.1 Conceptos fundamentales de los modelos NER . . . 23
  - 4.1.2 Categorización de las Entidades Sensibles . . . 24
  - 4.1.3 Etiquetado de Datos . . . 25

4.1.4 Generación de Datos sintéticos 26
4.1.5 Selección y Justificación del Modelo 27
4.1.6 Entrenamiento del Modelo 27
4.1.7 Conclusiones: Ventajas y Desventajas del enfoque NER . . . 28
4.2 Evaluación de modelos de IA locales 28
4.2.1 Selección del modelo: LLAMA3.2 3B 29
4.2.2 Instalación del modelo en local 29
4.2.3 Integración con backend FastAPI y Funcionamiento del sistema 31
4.2.4 Conclusiones: Ventajas y desventajas del enfoque local . . . 32
4.3 Enfoque híbrido basado en patrones (regex y bases de datos) 33
4.3.1 Diseño de la arquitectura 33
4.3.2 Conclusiones: ventajas y desventajas del enfoque híbrido . . 34
4.4 Despliegue del sistema de anonimización en AWS 35
4.4.1 Introducción al uso de AWS en el proyecto 35
4.4.2 AWS Bedrock: arquitectura y garantías de privacidad 36
4.4.3 Funcionamiento del modelo de detección y anonimización . . 36
4.4.4 Procesamiento interno en AWS Lambda 39
4.4.5 Paralelización del procesamiento y división en chunks 41
4.4.6 Integración de la función Lambda con Api Gateway 43
4.4.7 Costes asociados y latencia del sistema 44
4.4.8 Selección del modelo y rendimiento 46
4.5 Integración final en Word y descripción del funcionamiento del AddIn 47
4.5.1 Lógica interna del proceso de extracción de párrafos y anonimización del documento 48
4.5.2 Lógica interna del módulo de interacción con el modelo de lenguaje 51

5 Sistema de Contract Life Management (CLM) 55
5.1 Conceptos Básicos 55
5.1.1 Retrieval Augmented Generation (RAG) 55
5.1.2 Knowledge Graph (KG) 57
5.2 Diseño del Knowledge Graph 58
5.2.1 Generación de representaciones vectoriales (embeddings) . . 59
5.2.2 Cálculo de distancias semánticas 60
5.2.3 Etiquetado semántico automático de grupos de cláusulas . . 61
5.2.4 Detección de relaciones semánticas entre grupos contractuales 62
5.2.5 Generación de metadatos unificados por cláusula contractual 64
5.2.6 Generación y visualización del grafo de conocimiento contractual 65

5.3 Conclusión: Diseño e implementación de un grafo de conocimiento para contratos EPC 67
5.4 Diseño del Pipeline y generación de domain-specific RAG 68
5.4.1 Retriever: recuperación híbrida informada por grafo de conocimiento 68
5.4.2 Generator: generación de eventos estructurados a partir de contexto legal 70
5.4.3 Verifier: verificación de eventos mediante citas textuales . . 71
5.4.4 Orquestación de la arquitectura RAG + KG 73
5.4.5 Visualización interactiva del calendario contractual 74
5.5 Conclusión: integración del grafo de conocimiento en un sistema RAG especializado 75

6 Análisis de resultados **77**
6.1 Evaluación del rendimiento del anonimizado 77
6.1.1 Ejemplo de caso real 77
6.1.2 Resultados observados 78
6.1.3 Conclusión 80
6.2 Evaluación de resultados: grafo de conocimiento aplicado a contratos EPC 81
6.2.1 Agrupación semántica 81
6.2.2 Extracción de relaciones 81
6.2.3 Visualización interactiva 82
6.2.4 Observaciones y valoración 82
6.2.5 Conclusiones 83
6.3 Evaluación de resultados: generación del calendario contractual . . 83
6.3.1 Extracción de fechas y plazos 83
6.3.2 Clasificación de eventos 84
6.3.3 Referencias y citas 85
6.3.4 Visualización interactiva 85
6.3.5 Observaciones 85
6.3.6 Conclusiones 86
6.4 Contratos EPC como caso de uso: complejidad y aplicabilidad en otros contextos 86

7 Conclusión **87**
7.1 Objetivos cumplidos 87
7.2 Necesidad real 88
7.3 Proyecto de ingeniería 88

8 Líneas futuras **91**
8.1 Evolución hacia una plataforma modular y profesional 91
8.1.1 Evolución del módulo de anonimización 91
8.1.2 Evolución del módulo CLM 92
8.1.3 Convergencia: hacia una plataforma modular y personalizable 92
8.2 Perspectiva de futuro en el contexto de la evolución de la inteliencia artificial 93
8.2.1 Proyección comercial y desarrollo de colaboraciones 93

Anexo **95**

9 Anexo A: Diagramas OPM con descripción OPL **95**

10 Anexo B: Links a repositorios y demos **103**

Bibliografía **105**

List of Figures

1 Diagrama OPM Sistema de Anonimización vi
2 PII Marcado vii
3 PII Extraído vii
4 Diagrama OPM del CLM viii
5 OPM Diagram of the Anonymization System x
6 Marked PII xi
7 Extracted PII xi
8 OPM Diagram of the CLM System xii

4.1 Realiza llamada a API local de Ollama para poner a prueba modelo Llama3.2 30
4.2 Sistema de anonimización representado mediante metodología OPM 48

5.1 Arquitectura CLM representada mediante metodología OPM 56

6.1 PII Marcado 78
6.2 PII Extraído 79
6.3 PII Restaurado 79
6.4 Fragmento del KG compuesto por grupos no interconectados 81
6.5 Fragmento del KG compuesto por grupos relacionados 82
6.6 Nodo (grupo semántico) con meta información sobre chunks asociados 82
6.7 Vista General del Calendario Contractual 84

9.1 Diagrama OPM Arquitectura de anonimización 96
9.2 Diagrama OPM Arquitectura CLM 99

List of Tables

4.1 Descripción de los campos devueltos por el endpoint de Ollama . . 31

6.1 Entidades sensibles detectadas y su anonimización 80
6.2 Tipos de eventos contractuales clasificados 84

Listings

4.1 Servicio de integración local con Ollama en Python 32
4.2 Generación del prompt para datos personales 39
4.3 Combinación de resultados detectados 39
4.4 Procesamiento concurrente en ambas fases 41
4.5 Inicialización en Word 49
4.6 Extracción de todos los párrafos 49
4.7 Extracción del fragmento seleccionado 49
4.8 Procesamiento de la respuesta 50
4.9 Resaltado de PII 50
4.10 Sustitución de PII 51
4.11 Reversión de PII 51
4.12 Inicialización de la extensión 51
4.13 Extracción de texto seleccionado 52
4.14 Estructura de plantillas por idioma 52
4.15 Llamada al endpoint LLM 52
4.16 Inserción de respuesta en Word 53
5.1 Generación de chunks 59
5.2 Generación de Embeddings 59
5.3 Multiples hilos 60
5.4 Cálculo de similitud coseno 60
5.5 Agrupamiento con HDBSCAN 60
5.6 Prompt especializado para generación de etiquetas 61
5.7 Llamada a modelo LLM para etiquetado 62
5.8 Asignación de etiquetas a grupos 62
5.9 Filtrado por umbral de similitud 63
5.10 Selección de fragmentos relevantes 63
5.11 Instrucción para generación de relaciones 64
5.12 Asignación de grupos a fragmentos 64
5.13 Guardado de meta etiquetas 65
5.14 Tratamiento visual de loneliners 65
5.15 Cálculo de embedding para nodos del grafo 66

5.16 Resumen estructurado del grafo 66
5.17 Cálculo del embedding de grupo 69
5.18 Selección de cláusulas individuales relevantes 69
5.19 Expansión mediante el grafo de relaciones 69
5.20 Formateo del contexto 70
5.21 Construcción del prompt con Jinja2 70
5.22 Parseo robusto de la respuesta LLM 71
5.23 Evento contractual estructurado 71
5.24 Asignación de puntajes por coincidencia 72
5.25 Recuperación semántica contextualizada 73
5.26 Generación del calendario contractual 73
5.27 Validación de evento antes de su visualización 74

Acrónimos

ICAI	Insitituto Católico de Artes e Industrias
AWS	Amazon Web Services
CLM	Contract Life Management
NLTK	Natural Language Toolkit
PFC	Proyecto Fin de Carrera
PII	Personal Identifiable Information
PLN	Procesamiento de Lenguaje Natural
EPC	Engineering, Procurement and Construction
LLM	Large Language Model
RAG	Retrieval Augmented Generation
KG	Knowledge Graph

Introducción

1.1 Contexto

La evolución del Procesamiento de Lenguaje Natural (PLN) se ha consolidado como una de las áreas más influyentes y complejas dentro del campo de la inteligencia artificial, con un impacto creciente en sectores estratégicos como el jurídico. Esta disciplina se centra en el desarrollo de sistemas capaces de analizar, comprender y generar lenguaje humano, una tarea especialmente desafiante debido a la ambigüedad, riqueza y variabilidad propias de las lenguas naturales, en contraste con la precisión característica de los lenguajes formales.

Desde sus inicios en la década de 1950, el PLN ha transitado desde enfoques basados en reglas gramaticales explícitas hacia modelos estadísticos y, más recientemente, hacia técnicas de aprendizaje profundo. Estos avances han permitido abordar tareas de comprensión y generación textual a gran escala. Como apuntan Bird, Klein y Loper [17], el PLN abarca desde aplicaciones básicas, como el conteo de frecuencias de palabras, hasta sistemas complejos capaces de "entender" y responder de manera coherente a enunciados en lenguaje natural. Esta diversidad metodológica ha dado lugar a herramientas potentes que permiten desde la extracción de información y el análisis sintáctico, hasta la clasificación automática de textos o el reconocimiento de entidades nombradas.

El desarrollo y la consolidación del PLN han sido posibles, en parte, gracias a la disponibilidad de grandes corpus textuales, el incremento en la capacidad computacional y la creación de bibliotecas especializadas como NLTK. Estos factores han democratizado el acceso a técnicas avanzadas y han facilitado la integración de enfoques lingüísticos y computacionales en aplicaciones reales.

Por otra parte, la irrupción de los Modelos de Lenguaje de Gran Tamaño (LLMs, por sus siglas en inglés) ha supuesto un verdadero punto de inflexión en la evolución del PLN. Modelos como GPT (OpenAI), BERT (Google) o LLaMA (Meta) destacan por su capacidad para manejar miles de millones de parámetros

y procesar enormes volúmenes de datos textuales, lo que les permite generar y comprender lenguaje con una precisión y coherencia sin precedentes.

Este salto cualitativo fue posible gracias a hitos técnicos como la introducción de la arquitectura Transformer en 2017, que revolucionó el análisis del contexto y las relaciones entre palabras dentro de un texto. A diferencia de los modelos secuenciales previos, los Transformers emplean mecanismos de atención [51] que permiten captar el significado global de una oración o documento, lo cual resulta esencial para tareas complejas de comprensión y generación lingüística.

En el ámbito jurídico, los avances en Procesamiento de Lenguaje Natural han impulsado significativamente el crecimiento del LegalTech [27], donde estas tecnologías se han convertido en un pilar fundamental para la automatización de tareas como la creación de contratos, la gestión documental o la búsqueda de documentación inteligente. La integración de los LLMs en herramientas jurídicas no solo mejora la eficiencia operativa, sino que también refuerza la trazabilidad, la seguridad y el cumplimiento normativo, aspectos esenciales en la protección de datos y la gestión del riesgo legal.Esto no solo representa un aumento en la eficiencia y las capacidades de los profesionales del derecho, sino que también marca un punto de inflexión en su proceso formativo. La incorporación de estas tecnologías exige, desde la etapa universitaria, el desarrollo de un pensamiento crítico que permita a los estudiantes de derecho involucrarse activamente en la comprensión profunda de los contenidos teóricos que estudian.

En conclusión, la consolidación de los LLMs marca el inicio de una nueva etapa en el desarrollo del PLN, en la que la inteligencia artificial no solo asiste al profesional jurídico, sino que redefine su labor, abriendo paso a innovaciones que, hasta hace poco, parecían fuera del alcance de la tecnología.

1.2 Motivación y Alcance

Durante los últimos dos años, el Observatorio de LegalTech de ICADE, en colaboración con Garrigues, ha fomentado la colaboración entre alumnos de ICAI e ICADE a través de actividades híbridas tecno-legales como charlas, mesas redondas, eventos sobre tecnología e inteligencia artificial y hackathones. Estas iniciativas han tenido como objetivo principal introducir y formar a los estudiantes en áreas que tradicionalmente se perciben como opuestas, como el ámbito jurídico y el tecnológico. Durante este periodo, el fenómeno "ChatGPT" ha transformado significativamente la forma de estudiar, aprender e interactuar en diversos contextos educativos. Asimismo, se ha observado una tendencia creciente hacia la integración de herramientas de inteligencia artificial en el entorno empresarial, tanto a nivel nacional como internacional, con el propósito de agilizar los flujos de trabajo y dotar a los empleados de recursos que incrementen la eficiencia y

productividad. En este marco, el Trabajo de Fin de Grado se plantea como una solución orientada a aplicar herramientas avanzadas de inteligencia artificial al ámbito de la gestión contractual, una necesidad recurrente en sectores como el energético y el jurídico. El proyecto tiene como objetivo principal el desarrollo de un sistema que, mediante modelos generativos y técnicas de procesamiento de lenguaje natural, permita analizar, anonimizar y gestionar contratos de manera eficiente, contribuyendo a la optimización de procesos, la reducción de errores y el ahorro de tiempo en las operaciones empresariales.

La selección de los sistemas desarrollados en el presente trabajo no fue producto de la imaginación, sino el resultado de un proceso de consulta activa con profesionales del ámbito jurídico, incluidos abogados en ejercicio y expertos en tecnologías aplicadas al derecho. A lo largo de estas conversaciones se identificaron de forma detallada las problemáticas más recurrentes y las ineficiencias operativas que enfrentan en su día a día, tanto en despachos como en departamentos legales de empresa. Asimismo, se puso de manifiesto la insuficiencia de muchas de las herramientas actualmente disponibles, ya sea por su falta de precisión, su escasa personalización o su limitada capacidad para integrarse en flujos de trabajo reales.

Este intercambio directo con quienes viven de primera mano los retos del sector permitió adoptar de forma completa el papel de *Computer Scientist* tal y como lo define F. Brooks [20] y establecer una base sólida para definir el enfoque del Trabajo de Fin de Grado, orientándolo no solo hacia la aplicación de tecnologías emergentes, sino hacia la creación de soluciones concretas, alineadas con las verdaderas necesidades del entorno legal. En este sentido, el objetivo no fue únicamente diseñar sistemas técnicamente viables, sino también garantizar que respondieran a los problemas reales tal y como los propios profesionales desearían que fuesen resueltos: con precisión, eficiencia, integrabilidad y facilidad de uso. De esta forma, se buscó que el desarrollo tecnológico mantuviera una estrecha conexión con la práctica jurídica cotidiana, aportando valor añadido desde una perspectiva tanto técnica como funcional.

1.3 Objetivos

El presente trabajo se estructura en torno al desarrollo de dos sistemas complementarios aplicados a la gestión contractual en el contexto de contratos EPC (Engineering, Procurement and Construction). El primero de ellos consiste en un sistema de Contract Lifecycle Management (CLM) especializado en contratos EPC, que abarca tanto la fase de creación como la de seguimiento del contrato.

En una primera etapa, el enfoque se centrará en el desarrollo de un sistema de anonimización de contratos que permita generar copias editables de los documentos contractuales cumpliendo con el Reglamento General de Protección de Datos

(GDPR). Para ello, se eliminarán o sustituirán datos sensibles como nombres, direcciones, identificadores personales, fechas, teléfonos, cuentas bancarias y datos fiscales, garantizando así la posibilidad de compartir dichos documentos de forma segura y ágil.

Los objetivos principales de este segundo módulo incluyen:

- El desarrollo de un add-in para Word que integre el proceso de anonimización.
- La implementación de un mecanismo de verificación de datos sensibles, haciendo uso de modelos de lenguaje de gran tamaño (LLMs) ejecutados en la nube garantizando la privacidad de estos datos mediante infrastructuras AWS.

En una segunda etapa, el proyecto contempla la generación automática de un calendario con los hitos principales del contrato, incluyendo recordatorios y alertas, lo cual permitirá realizar un seguimiento eficiente y libre de riesgos.

Entre los objetivos principales de este módulo CLM se encuentran:

- La creación de una arquitectura que combina *RAG* y *Knowledge Graphs* para la generación de un calendario contractual especializado en contratos de tipo EPC (Engineering, Procurement & Construction)
- La visualización del calendario contractual con una interfaz interactiva y adaptable a diferentes contratos.

1.4 Metodología

El enfoque metodológico de este Trabajo de Fin de Grado (TFG), centrado en el desarrollo de un sistema innovador para la gestión de contratos mediante inteligencia artificial, se basa en los principios y la filosofía de Y Combinator (YC). Esta metodología se caracteriza por su énfasis en la interacción constante con los clientes, el desarrollo ágil de productos adaptados a sus necesidades y la iteración continua basada en datos y retroalimentación real. Los pilares fundamentales que guían este proyecto son los siguientes:

Comunicación directa y constante con los clientes

La base de la metodología YC reside en resolver problemas reales para usuarios específicos. Por ello, un componente clave del desarrollo del TFG consiste en mantener una comunicación continua con los potenciales usuarios del sistema:

- **Entrevistas iniciales con clientes:** Desde el inicio del proyecto, se llevan a cabo entrevistas con profesionales del sector energético, legal y de contratos EPC para identificar necesidades concretas y puntos críticos en sus flujos de trabajo.

- **Validación temprana del concepto:** Antes de desarrollar el Producto Mínimo Viable (MVP), se presentan pruebas de concepto (POC) y flujos de trabajo preliminares a los usuarios para validar la dirección del diseño.

- **Feedback recurrente:** Durante todo el desarrollo, se mantiene un ciclo continuo de reuniones con los clientes para evaluar funcionalidades implementadas y recoger sugerencias de mejora.

Esta comunicación no solo garantiza que el producto responda a necesidades reales, sino que también fomenta su adopción al implicar a los usuarios en el proceso de creación.

Desarrollo ágil y rápido adaptado a las necesidades del cliente

El proyecto sigue un enfoque de desarrollo iterativo rápido, con el objetivo de maximizar el valor generado desde las primeras fases:

- **Producto Mínimo Viable (MVP):** En las primeras etapas se implementa un MVP con funcionalidades esenciales, como la identificación de datos confidenciales y la anonimización básica de contratos. Este prototipo permite recopilar retroalimentación temprana y validar su viabilidad en entornos reales.

- **Iteraciones priorizadas:** Cada nueva versión del producto aborda los problemas más relevantes identificados por los usuarios, garantizando una evolución alineada con sus necesidades.

- **Entrega continua:** Se adopta un modelo de entregas periódicas para mantener a los usuarios involucrados en el desarrollo, permitiendo evaluaciones constantes e incorporación ágil de mejoras.

Iteración constante y aprendizaje útil

La filosofía YC prioriza el aprendizaje rápido a través del feedback, evitando invertir excesivamente en perfeccionar el producto sin antes probarlo. Este principio se traduce en las siguientes acciones:

- **Análisis de métricas clave:** Indicadores como el tiempo de procesamiento de contratos, la precisión de la anonimización y el nivel de satisfacción del usuario se emplean para medir el impacto de cada iteración y guiar las mejoras.

- **Ciclos de mejora continua:** Cada iteración incorpora aprendizajes previos, ajustando funcionalidades y experiencia de usuario para optimizar progresivamente el sistema.

- **Pruebas de usabilidad:** Se realizan evaluaciones periódicas para detectar fricciones y oportunidades de mejora, permitiendo priorizar tareas de desarrollo en función de los resultados obtenidos.

Este enfoque tiene como objetivo final ofrecer una solución robusta, útil y alineada con las necesidades del mercado, evitando desarrollar funcionalidades que no aporten un valor significativo.

Estado del Arte

2.1 Expansión de las Tecnologías LegalTech

En los últimos años se ha producido un boom de tecnologías legales (LegalTech) que prometen revolucionar el trabajo jurídico automatizando tareas y ahorrando horas de trabajo. Sin embargo, muchas de estas herramientas han enfrentado dificultades para integrarse en la rutina diaria de los abogados. Un problema frecuente es la falta de integración con suites ofimáticas ampliamente usadas como Microsoft Office 365 – se estima, por ejemplo, que Microsoft Teams (parte de Office 365) era utilizada por el 79 por ciento[1]. Muchas soluciones se ofrecen como plataformas separadas en lugar de integrarse con las herramientas existentes, dificultando su adopción práctica. A continuación, se analizan las herramientas más destacadas en dos categorías clave: los gestores de ciclo de vida de contratos (CLM, Contract Lifecycle Management) y las herramientas de anonimización de documentos con datos sensibles.

2.1.1 Contract Life Management en el mercado nacional

En el ámbito de la gestión contractual, han emergido en España varias empresas pioneras. Bigle Legal [16], por ejemplo, es una startup catalana que ha recaudado varios millones de euros en financiación y ofrece dos herramientas destacadas enfocadas en contratos. Su módulo de Contract Management permite centralizar la creación, edición y almacenamiento de contratos, con plantillas personalizables y controles de acceso que garantizan consistencia y precisión. También facilita la colaboración en la revisión y aprobación de contratos, optimizando el flujo de trabajo entre equipos y reduciendo errores. Por otro lado, su módulo de Contract Lifecycle Management (CLM) abarca todo el ciclo de vida contractual: desde la generación inicial hasta la firma, seguimiento de plazos, renovaciones y finalización. Esta herramienta automatiza alertas de fechas clave para prevenir incumplimientos o retrasos, asegurando una gestión eficiente en todas las fases del contrato.

Bigle Legal ha apostado por la integración; por ejemplo, ofrece integraciones con plataformas de terceros (como Microsoft iManage) para insertarse en los flujos de trabajo corporativos existentes. Recientemente también incorporó Bigle Libra, un asistente de IA jurídica integrado en su CLM, enfocado en confidencialidad y reducción de alucinaciones en comparación con herramientas generalistas.

Otra empresa española notable es LexDoka [33] (anteriormente referida como LexDoca). LexDoka ofrece una plataforma avanzada de gestión de documentos y contratos, diseñada para automatizar y optimizar cada fase del ciclo de vida contractual. Permite crear contratos rápidamente a partir de plantillas personalizables, garantizando la consistencia y el cumplimiento normativo en cada documento. Su plataforma facilita la negociación en línea de los contratos mediante herramientas colaborativas con trazabilidad total de cambios, comentarios y aprobaciones, acelerando el proceso de acuerdo. Además, incorpora firma electrónica integrada, eliminando la necesidad de soluciones de firma externas. Todos los documentos se almacenan en un repositorio central seguro con capacidades de búsqueda semántica, etiquetado y alertas inteligentes, lo que brinda control eficiente, organización clara y acceso inmediato a la documentación legal desde una única interfaz. LexDoka incluso cuenta con módulos impulsados por IA, como LexAnalyzer, que detecta inconsistencias, resume cláusulas y extrae datos clave de los contratos, y un asistente virtual (Lexy) que ayuda a los usuarios a gestionar obligaciones contractuales. Esta propuesta integral ha posicionado a LexDoka como una de las plataformas LegalTech más innovadoras de 2025 en España [23].

2.1.2 Contract Life Management en el mercado internacional

A nivel internacional, el mercado de Contract Lifecycle Management ha madurado con varias soluciones líderes en los últimos tres años, muchas de ellas potenciadas por IA. A continuación, se resumen algunas plataformas destacadas:

- Icertis Contract Management[28]: Pionera en CLM corporativo, ampliamente adoptada por empresas globales. Ofrece gestión integral del ciclo de contratos con potente motor de búsqueda y analítica post-firma. Icertis ha incorporado módulos de analytics impulsados por IA para monitorizar riesgos y cumplimiento en grandes volúmenes de contratos.

- DocuSign CLM[21]: Resultado de la adquisición de SpringCM por DocuSign, integra la funcionalidad de gestión de contratos con la firma electrónica líder de mercado. Permite automatizar todas las etapas del ciclo de vida contractual (creación, negociación, firma y renovación) y recientemente añadió

funciones inteligentes de acuerdos (Smart Agreements) con IA para extraer metadatos y obligaciones clave.

- Ironclad[29]: Plataforma de CLM enfocada en experiencia de usuario y colaboración. Es reconocida por su interfaz simple y segura para crear y negociar contratos en línea. Ironclad ha incorporado IA generativa (modelo AI Assist) para revisar cláusulas y sugerir redacciones, agilizando el ciclo de contratación.

- Evisort[25]: Solución de gestión contractual end-to-end nativa en IA. Evisort destaca por incluir un modelo de lenguaje propio entrenado específicamente en lenguaje jurídico, capaz de revisar contratos y responder preguntas sobre su contenido. Automatiza la clasificación de contratos y la extracción de cláusulas importantes, ayudando a departamentos legales a encontrar rápidamente información en sus repositorios.

- SirionLabs[48]: Enfocada en la fase post-firma, SirionLabs ofrece un CLM con analítica de cumplimiento contractual impulsada por IA. Su módulo de AI-powered analytics identifica desviaciones en el cumplimiento de obligaciones y fechas, y ayuda a gobernar los contratos una vez activos.

2.1.3 Soluciones para la Anonimización de Datos Sensibles

En el procesamiento de documentos legales con información sensible, la anonimización y detección de datos personales es un tema crucial. En España destaca la empresa vasca Nymiz, que ofrece una solución de anonimización de datos basada en inteligencia artificial. El software de Nymiz es capaz de identificar datos personales y sensibles en grandes volúmenes de documentos (Word, PDF, Excel, etc., incluso bases de datos) y luego proceder a su anonimización o seudonimización. Esto incluye eliminar o enmascarar directamente la información (p. ej., reemplazar nombres por una franja negra o asteriscos) o sustituir los datos por valores sintéticos/pseudónimos mediante tokenización. Por ejemplo, un nombre puede ser detectado y reemplazado por un código aleatorio, cumpliendo así con la ley de protección de datos, ya que tras la anonimización el documento deja de contener datos personales reales. La herramienta de Nymiz utiliza técnicas de Procesamiento de Lenguaje Natural (NLP) entrenadas para reconocer patrones contextuales (como dos palabras en mayúsculas seguidas que podrían ser nombre y apellido) en múltiples idiomas, simulando la forma en que un humano identificaría datos personales. Esto le permite localizar información sensible en diferentes idiomas y contextos con alta precisión, aunque siempre existe un margen de error al ser tecnologías emergentes.

Un punto fuerte de Nymiz es que cumple con normativas de privacidad como GDPR y HIPAA, proporcionando diferentes modos de anonimización (reversible o irreversible) según los requisitos legales. De hecho, una vez anonimizados los documentos, estos quedan exentos de la aplicación del RGPD al ya no contener PII (Personally Identifiable Information), lo que permite a las empresas analizar y compartir esos datos con menor riesgo. Nymiz se integra fácilmente en sistemas existentes mediante una solución SaaS y API, facilitando su adopción en despachos legales, sector financiero, sanitario, etc. Cabe mencionar que la tecnología de Nymiz se inspira en desarrollos semánticos previos de su empresa matriz (Serikat) con más de 18 años de experiencia en enmascaramiento de datos para el sector jurídico, aportando solidez a la solución.

A nivel internacional, existen numerosas herramientas de anonimización y protección de datos personales, incluyendo tanto soluciones comerciales como de código abierto. Entre las herramientas comerciales de última generación se encuentran:

- Syntho (Países Bajos)[49]: plataforma que combina un motor de generación de datos sintéticos con un escáner impulsado por IA para detectar PII/PHI en sistemas y bases de datos. Permite a las organizaciones reemplazar datos reales por datos sintéticos realistas, ofreciendo desidentificación inteligente manteniendo la utilidad estadística de los datos. Es utilizada en sectores como salud, finanzas y logística para crear conjuntos de datos simulados que preservan patrones pero sin información identificable.

- K2View Data Masking (EE.UU.)[30]: solución empresarial que anonimiza datos directamente en bases de datos y archivos. Ofrece cientos de funciones de enmascaramiento (sustitución, barajado, cifrado, etc.) y soporte para tokenización. Su fortaleza es la integración con entornos corporativos complejos: puede conectarse a múltiples bases de datos, sistemas heredados y flujos CI/CD, aplicando enmascaramiento continuo en tiempo real. K2View mantiene la integridad referencial de los datos (p. ej., si se enmascara un ID en dos tablas relacionadas, conserva la correlación) y soporta generación de datos sintéticos para rellenar entornos de prueba.

- Broadcom CA Test Data Manager[19]: suite enfocada en pruebas de software que incluye capacidades avanzadas de anonimización. Proporciona redacción automática de datos sensibles, tokenización y generación de datos sintéticos con control granular. Sus APIs abiertas permiten integrarlo en pipelines DevOps, asegurando que en cada entorno de desarrollo/pruebas los datos estén desidentificados de forma consistente. Es valorado por su flexibilidad de licenciamiento (planes escalables según tamaño de empresa) aunque la

configuración inicial puede ser compleja, mitigada por un soporte técnico sólido.

Además de estas, existen herramientas de código abierto reconocidas, como ARX[13], desarrollada en contextos académicos/UE, que implementa técnicas clásicas de anonimización (supresión, k-anonimato, l-diversidad, etc.). ARX es una herramienta open source potente que soporta múltiples modelos de privacidad y transformaciones, permitiendo analizar la calidad de los datos anonimizados mediante métricas de pérdida de información. Su uso está dirigido a instituciones académicas o pequeñas organizaciones por no tener costo de licencia, aunque requieren conocimientos técnicos para su correcto uso y no integran AI avanzada como las soluciones comerciales mencionadas.

2.2 Estado actual de LLMs, RAG y Sistemas de Agentes

2.2.1 Competencia Reciente en LLMs

El ritmo de avances en Large Language Models (LLMs) ha sido vertiginoso. No pasa una semana sin que se anuncie un nuevo modelo de lenguaje natural o una mejora sustancial de los existentes. Las grandes tecnológicas compiten por modelos más capaces, buscando liderar el mercado con mejores prestaciones y especializaciones. OpenAI popularizó los LLM con GPT-3 y GPT-4, desplegados comercialmente via API, desencadenando una carrera donde participan Anthropic (Claude 2 y sucesores), Google (PaLM 2 y el esperado Gemini), Meta (LLaMA 2 y 3, modelos abiertos), Microsoft (integrando GPT-4 en sus productos) y más recientemente Mistral AI con su familia de modelos especializados. Esta compañía francesa ha desarrollado una estrategia diferenciada centrada en modelos tanto abiertos como propietarios para entornos empresariales, donde la flexibilidad de despliegue, eficiencia y control son fundamentales. Un ejemplo representativo es el lanzamiento de Mistral Large 2 en 2024, posicionado como modelo de frontera para tareas de alta complejidad empresarial. La familia Mistral incluye desde modelos edge como Ministral 3B y 8B optimizados para dispositivos móviles y laptops, hasta Pixtral Large como modelo multimodal de frontera[38]. Mistral Large 2 está equipado con capacidades mejoradas de function calling y retrieval, entrenado para ejecutar llamadas de función tanto paralelas como secuenciales, lo que lo convierte en motor de aplicaciones empresariales complejas. Es notable que Mistral ofrece capacidades empresariales que incluyen despliegue híbrido o on-premises/in-VPC, entrenamiento personalizado post-training, e integración con sistemas y herramientas empresarialea. Los modelos Mistral integran nativamente

function calling[37] para conectar con herramientas externas, soporte para RAG con embeddings especializados, y outputs estructurados, reflejando la tendencia hacia LLM preparados para flujos agentivos y aplicaciones empresariales que requieren interacción confiable con sistemas externos y mantenimiento de control total sobre la infraestructura de despliegue.

Desde el punto de vista académico, han surgido trabajos clave que marcan el estado del arte. Por ejemplo, el concepto de Retrieval-Augmented Generation (RAG)[32] demuestra cómo combinar un modelo pre-entrenado con una base de conocimiento externa consultable podía mejorar drásticamente tareas de pregunta-respuesta abierta. Asimismo, se han investigado métodos para dotar a los LLM de capacidad de usar herramientas: Toolformer[46] (propuesto por investigadores de Meta AI en 2023) entrenó un modelo para decidir por sí mismo cuándo llamar a APIs externas (calculadora, buscador, traductor, etc.) y cómo incorporar los resultados en su generación de texto. Este enfoque logró que un LLM mediano resolviera operaciones aritméticas y consultas de conocimiento de forma más eficaz, combinando sus habilidades lingüísticas con la exactitud de herramientas especializadas. La idea subyacente es que los modelos pueden aprender a orquestar herramientas para paliar sus limitaciones (por ejemplo, usar una calculadora para matemáticas precisas en lugar de confiar en su memoria entrenada). Otro avance fue el método ReAct (Yao et al., 2022), que introdujo un marco de razonamiento y acción donde el modelo genera explícitamente pensamientos intermedios y acciones (consultas a herramientas) de manera iterativa, mejorando la capacidad de resolver tareas complejas paso a paso. Todas estas innovaciones académicas han confluido en la práctica: los LLM actuales incorporan cada vez más estos paradigmas en productos comerciales. En resumen, el panorama de LLM en 2025 se caracteriza por: modelos cada vez más grandes pero optimizados (surgen también modelos abiertos más pequeños pero eficientes, como Mistral 7B, LLaMA-2 13B, etc.), enfoque en multimodalidad (ej. modelos que aceptan texto e imágenes, o que generan también código), y un fuerte acento en capacidades agentivas y de recuperación de información para aplicaciones de negocio. Las guías de desarrollo publicadas por los líderes del sector enfatizan buenas prácticas de construcción con estos modelos, como veremos a continuación.

2.2.2 Integración de información específica mediante Retrieval Augmented Generation (RAG)

Una de las técnicas más importantes hoy para implementar LLMs en entornos especializados (como el legal) es Retrieval-Augmented Generation (RAG), o generación aumentada con recuperación de información. RAG consiste en dotar al modelo de lenguaje de acceso a una base de conocimiento externa, típicamente a

través de un vector DB o índice de documentos, de modo que ante cada consulta primero se recuperan documentos relevantes y luego el modelo genera la respuesta apoyándose en ese contexto recuperado. En esencia, combina un componente de information retrieval (buscador) con el modelo generativo, logrando respuestas más precisas, actualizadas y con posibilidad de citar fuentes.

En aplicaciones jurídicas, RAG resulta sumamente valioso. Un LLM general (por ejemplo, GPT-4) puede tener conocimientos legales hasta cierta fecha, pero para asesorar con exactitud necesita la normativa y jurisprudencia vigente o los detalles de contratos específicos. Con RAG, el flujo típico sería: el usuario pregunta algo sobre un contrato o ley; el sistema busca en un repositorio de contratos o en una base de datos legal los párrafos pertinentes; esos textos se proporcionan al LLM en el prompt; y el modelo elabora su respuesta basándose en dicha información. Esto reduce al mínimo las "alucinaciones" y errores factuales del modelo, ya que está anclando sus respuestas en documentos reales[26]. Además, aporta transparencia, pues es posible devolver junto con la respuesta las citas o referencias de los documentos utilizados (lo cual es crítico en entornos legales para verificar fuentes).

Desde su aparición, RAG ha evolucionado hacia esquemas cada vez más sofisticados. Hoy se habla de Agentic RAG, donde intervienen agentes (LLMs autónomos) que pueden usar múltiples herramientas además de la simple recuperación de textos. En RAG tradicional, el proceso suele limitarse a recuperar pasajes de un vector DB y generar texto. En cambio, en Agentic RAG un agente de IA puede decidir usar múltiples herramientas: por ejemplo, primero buscar en una base documental, luego llamar a una calculadora para computar daños y perjuicios, o incluso delegar subtareas a otros agentes. La incorporación de agentes (de la cual hablaremos en la siguiente sección) expande RAG más allá del QA documental, permitiendo resolver tareas complejas que involucren diferentes tipos de acciones.

Implementar RAG eficazmente requiere seguir ciertas mejores prácticas. Las guías de OpenAI, Anthropic y otras compañías destacan recomendaciones como: segmentar correctamente los documentos (chunking) para su indexación, usar embeddings de calidad para las búsquedas vectoriales, e implementar mecanismos para resolver contradicciones entre la respuesta del modelo base y el contenido recuperado. Un desafío conocido es el conflicto de conocimiento: el LLM tiene conocimientos almacenados en sus parámetros (que podrían estar desactualizados o ser generales) y la documentación recuperada aporta conocimiento específico. Para manejar esto, se emplean indicaciones en el prompt que instruyen al modelo a dar prioridad a la información del documento proporcionado sobre su conocimiento previo. También se puede ajustar la temperatura del modelo baja para forzar que se apegue al contexto dado, o incluso usar enfoques de "context distillation" donde el modelo es afinado para confiar más en fuentes externas. OpenAI, por ejemplo,

sugiere técnicas de prompt engineering donde se le dice al asistente algo como: "Si la información adjunta contradice tu conocimiento, utiliza únicamente la información adjunta".

Otra recomendación es proveer al usuario las referencias. Empresas como Microsoft y OpenAI han lanzado complementos de ChatGPT que implementan RAG con citas automáticas, para fomentar la verificabilidad. Incluso se han creado benchmarks específicos como RAGBench para medir qué tan bien un modelo soporta este modo de trabajo[14].

En cuanto a frameworks para construir sistemas RAG, una de las bibliotecas más destacadas es LangChain[31]. LangChain proporciona componentes para gestionar prompts, vectores, conectores a bases de datos y orquestación de agentes, lo que facilita crear aplicaciones donde un LLM responde basándose en documentos externos. También existen otras, como LlamaIndex (GPT Index), que simplifica la indexación de documentos y consulta con LLMs, o herramientas nativas en la nube (Azure Cognitive Search, API de búsqueda de Google PaLM, etc.). Siguiendo la estela, OpenAI ha introducido recientemente su Assistants API con soporte de retrieval plugins, que automatiza parte del proceso de RAG en su ecosistema.

2.2.3 Agentes de IA y Function Calling

Conforme los LLMs han mejorado en comprensión y razonamiento, ha surgido el paradigma de los Agentes de IA: sistemas impulsados por LLM capaces de operar autónomamente, descomponer tareas, utilizar herramientas externas y encadenar pasos para lograr un objetivo. Un agente típico inicia con una instrucción u objetivo dado por el humano, tras lo cual el propio agente planifica la secuencia de acciones necesarias, interactuando con su entorno (llamando APIs, buscando datos, ejecutando código) y ajustando su plan en función de los resultados que obtiene[47].A diferencia de un simple chatbot que responde en lenguaje natural, un agente puede actuar: por ejemplo, al recibir el mandato "obtener todos los plazos de este contrato y generar un calendario de alertas", un agente legal podría buscar en el contrato las fechas relevantes, luego llamar a una API de calendario para crear eventos, y finalmente confirmar al usuario que la tarea fue realizada.

Desde un punto de vista técnico, un agente es esencialmente un LLM operando en un bucle de percepción-cálculo-acción. En cada iteración, el agente evalúa el estado (por ejemplo, la última instrucción del usuario o el resultado de la última acción), decide qué herramienta (si alguna) usar a continuación, o si ya puede dar una respuesta final. Este enfoque ha sido formalizado en varios marcos de referencia. Anthropic, en su guía de Building Effective Agents, describe patrones de diseño como orchestrator-workers (un LLM orquestador delegando subtareas a otros LLMs trabajadores) y evaluator-optimizer (un LLM generador y otro que evalúa y retroalimenta, refinando la respuesta en bucle). Dichos patrones pueden

combinarse para lograr agentes muy sofisticados. Pero un caso particular es el agente autónomo monolítico que iterativamente hace razonamiento y llamadas de herramienta, también conocido como el patrón ReAct anteriormente citado.

OpenAI ha incorporado el soporte para agentes de forma explícita en sus APIs. En 2023 presentó la funcionalidad de Function Calling[41] en modelos como GPT-4, que permite definir funciones (herramientas) que el modelo puede invocar cuando lo considere necesario. Básicamente, el desarrollador proporciona descripciones de funciones (por ejemplo, buscarcontrato(fecha), enviaremail(destinatario, mensaje)) y el modelo, si lo cree oportuno durante una conversación, responde con una llamada a una de esas funciones en lugar de texto libre. Esta fue la base de los Plugins de ChatGPT también. Por ejemplo, con function calling GPT-4 puede buscar información en la web, calcular algo, o extraer datos estructurados, todo siguiendo las mismas instrucciones en lenguaje natural del usuario[42]. Anthropic, por su lado, ha mantenido un enfoque prudente pero también posibilita agentes: Claude tiene contexto extensísimo (100k tokens) que permite incluir documentación y resultados intermedios, y si bien no expuso una API de herramientas tan pronto como OpenAI, sus guías recomiendan patrones similares de encadenamiento de prompts y uso de APIs intermedias supervisadas por el desarrollador. Meta, además del mencionado Toolformer, ha fomentado agentes a través de la comunidad open source: con LLaMA disponible, surgieron proyectos como AutoGPT, BabyAGI, LangChain Agents, etc., que aprovecharon modelos abiertos para crear agentes autónomos con diversos grados de éxito.

Es importante señalar que los agentes de IA en producción hoy suelen ser implementados con límites y monitorización. Las empresas tecnológicas aconsejan usar checkpoints o validaciones en los bucles de los agentes para evitar situaciones de bucle infinito o comportamientos no deseados. Por ejemplo, Anthropic sugiere incluir condiciones de parada (máximo número de iteraciones, o requerir confirmación humana en ciertos pasos) y guardrails[22] para prevenir que un agente malinterprete su rol. OpenAI, en su documentation de función, enfatiza definir claramente qué hace cada herramienta y manejar las excepciones cuando el modelo sale de contexto. En términos de mejores prácticas: diseñar cuidadosamente el conjunto de herramientas disponibles para el agente (menos es más, proveer solo lo necesario con interfaces claras), probar en entornos controlados (sandbox) antes de desplegar en vivo, y siempre preservar un grado de supervisión humana cuando se trate de automatizaciones críticas.

Pese a estos cuidados, los agentes están resolviendo tareas antes impensadas. Por ejemplo, agentes de código ya asisten en generación y depuración de software: Github Copilot X y similares actúan como agentes que pueden incluso ejecutar fragmentos de código para validar soluciones. En legal, podemos imaginar un agente que lea un contrato, detecte cláusulas confidenciales y automáticamente las

anonimize usando una herramienta de la categoría de Nymiz, luego tome el contrato anonimizado y lo resuma en puntos clave, y posteriormente genere un checklist de obligaciones en una tabla. Todo ello encadenado sin intervención humana directa, más que la instrucción inicial. Para lograr ese flujo, internamente habría varias herramientas: extracción de texto, anonimización, resumen, formateo, etc., y un agente orquestador (posiblemente construido con LangChain) que decide el orden y aplica cada sub-herramienta. Este tipo de agente orquestador encaja con el patrón orchestrator-workers de Anthropic y de hecho ya existen implementaciones abiertas (por ejemplo, HuggingGPT de Microsoft Research coordinaba múltiples modelos especializados bajo las órdenes de un LLM principal).

Introducción a la arquitectura del sistema

Las soluciones y tecnologías LegalTech tienen como objetivo automatizar tareas, optimizar los flujos de trabajo y mejorar la eficiencia del ejercicio profesional del derecho. Para alcanzar estos fines, resulta esencial identificar cuáles son los requisitos mínimos que deben cumplirse para que los despachos de abogados y las asesorías jurídicas se planteen siquiera la posibilidad de incorporar dichas herramientas. Bajo estas premisas, durante el proceso de generación y validación de ideas, surgieron de forma recurrente dos necesidades clave: las garantías en materia de privacidad de los datos y la integración directa de las herramientas dentro del entorno de Microsoft Word.

La principal razón por la que muchas soluciones emergentes fracasan en este sector es la falta de garantías sólidas de privacidad, lo que constituye una de las mayores preocupaciones de los despachos a la hora de permitir el uso de modelos comerciales como los desarrollados por ChatGPT, Anthropic o Meta. Si bien estas empresas destacan por el altísimo rendimiento de sus modelos de lenguaje (LLMs, por sus siglas en inglés) y su constante evolución, no ofrecen una transparencia total sobre el tratamiento de los datos que reciben a través de las interacciones con los usuarios. Esta falta de claridad genera importantes desafíos de cumplimiento normativo (compliance), lo que ralentiza considerablemente la adopción de estas tecnologías en el día a día del trabajo jurídico.

Con el fin de garantizar la privacidad desde las primeras fases del desarrollo, se orientó inicialmente el proyecto hacia la búsqueda de soluciones que pudieran ejecutarse en entornos locales, de manera que toda la información permaneciera dentro de la infraestructura de las propias organizaciones. Sin embargo, dichas soluciones no ofrecieron resultados suficientemente satisfactorios en términos de rendimiento. Por ello, la segunda fase del desarrollo consistió en una investigación exhaustiva sobre alternativas que permitieran ejecutar modelos de lenguaje con garantías adecuadas de privacidad. Esta etapa culminó con el diseño de un sistema

de reconocimiento de datos sensibles basado en LLMs, desplegado mediante el servicio AWS Bedrock, cuya implementación se detallará más adelante en este documento.

Por otra parte, muchas de las herramientas actualmente disponibles en el mercado requieren que los abogados trabajen en plataformas nuevas o poco familiares, lo cual introduce una barrera significativa. Mientras las empresas desarrolladoras prometen mayor eficiencia, en la práctica terminan desviando la atención de los profesionales jurídicos, quienes deben invertir tiempo en procesos de formación y adaptación. En este contexto, uno de los objetivos principales del presente proyecto ha sido el desarrollo de complementos (Add-ins) para Microsoft Word que minimicen o eliminen esta curva de aprendizaje, facilitando así una integración fluida en los flujos de trabajo habituales del profesional jurídico.

3.1 Estructura y Dessarollo de un AddIn de Word

3.1.1 Nucleo declarativo del AddIn: Manifest

El funcionamiento de un complemento (Add-in) para Microsoft Word se basa en una arquitectura web[36] cuya configuración principal se encuentra definida en un archivo denominado manifest[34]. Este archivo, que puede adoptar formato XML o JSON según el tipo de desarrollo, actúa como núcleo declarativo del Add-in, ya que especifica tanto su comportamiento como su integración dentro del entorno de Microsoft Office.

En primer lugar, el manifest contiene los metadatos básicos del complemento: nombre, identificador único (GUID), versión, descripciones, e información de localización para adaptarlo a distintos idiomas. A nivel funcional, este archivo establece cómo se integra el complemento en la interfaz de usuario de Word, permitiendo definir qué comandos aparecerán en la cinta de opciones (ribbon), si se habilitan pestañas personalizadas o botones específicos, y cómo se comporta el panel lateral (task pane), entre otros aspectos.

Otro aspecto clave del manifest es la declaración de los permisos necesarios para que el complemento interactúe con el documento. Estos permisos pueden limitarse, por ejemplo, a la lectura del contenido, o incluir también la escritura y modificación, dependiendo de la funcionalidad que se desee implementar. De este modo, el sistema puede controlar con precisión el alcance del complemento dentro del entorno de Office, lo cual resulta especialmente relevante desde una perspectiva de seguridad y cumplimiento normativo.

Asimismo, el archivo define las rutas a las que se conectará el Add-in, restringiéndolas a direcciones seguras que utilicen el protocolo HTTPS. Esta medida garantiza que todos los recursos cargados (como scripts, estilos o páginas web)

provengan de fuentes confiables, y es un requisito imprescindible para que el complemento pueda ejecutarse en el ecosistema de Office.

3.1.2 Estructura de Aplicación Web

Junto con el archivo manifest, todo complemento de Word necesita una aplicación web que actúe como interfaz visual y lógica del sistema. Esta aplicación suele estructurarse en una carpeta denominada src o, más específicamente, en un subdirectorio llamado *taskpane*, que contiene los recursos necesarios para desplegar el panel lateral interactivo dentro del propio documento de Word.

El componente central de esta interfaz es un archivo HTML, normalmente llamado taskpane.html, que define la estructura visual que verá el usuario al activar el complemento. A través de este archivo se construye el contenido del panel, que puede incluir formularios, botones, menús o cualquier otro elemento visual necesario para la interacción con el usuario [35].

Para aplicar estilos y garantizar una experiencia de uso coherente con la estética del entorno de Microsoft Word, se utiliza una hoja de estilos complementaria, *taskpane.css*. Este archivo permite controlar de forma independiente el diseño gráfico del complemento, separando las responsabilidades de presentación de la lógica funcional.

La funcionalidad del panel se implementa principalmente mediante un archivo JavaScript, comúnmente llamado *taskpane.js*. En este archivo se define el comportamiento dinámico de la aplicación, así como su capacidad para interactuar directamente con el documento de Word. Esta interacción se realiza a través de la API Office.js, una biblioteca JavaScript proporcionada por Microsoft que permite acceder y modificar el contenido del documento de manera segura y controlada. Para poder utilizar esta API, es necesario importar el script correspondiente desde el CDN oficial de Microsoft, habitualmente mediante una instrucción en el encabezado del HTML como la siguiente:

```
<script src="https://appsforoffice.microsoft.com/lib/1/hosted/office.js">
    </script>
```

Una vez incorporada esta biblioteca, el desarrollador puede hacer uso de las funciones específicas para Word que ofrece Office.js, como acceder al texto del documento, insertar contenido, leer propiedades del archivo o manipular controles de contenido. Este enfoque permite mantener el núcleo funcional del complemento dentro del navegador, sin necesidad de instalar software adicional, lo que favorece la compatibilidad entre plataformas y facilita el despliegue en entornos corporativos.

3.1.3 Archivos Estátios

Además de los archivos de configuración y de la aplicación web que constituye la interfaz del complemento, todo Add-in de Word incluye una serie de recursos estáticos que cumplen funciones tanto visuales como técnicas. Estos recursos, aunque no intervienen directamente en la lógica del sistema, son fundamentales para garantizar una integración coherente con el entorno de Word y una experiencia de usuario profesional.

Entre estos activos estáticos se encuentran, principalmente, los iconos y logotipos que se utilizan para representar visualmente el complemento en la cinta de opciones (ribbon) de Microsoft Word. Estos elementos gráficos, generalmente en formato PNG o SVG, deben estar alojados en ubicaciones accesibles mediante el protocolo HTTPS y deben ser referenciados explícitamente en el archivo manifest. De este modo, Word puede cargar las imágenes de forma segura y mostrarlas correctamente en los botones o comandos definidos por el complemento.

3.1.4 Servidor local para el desarrollo

Durante el proceso de desarrollo de un complemento para Microsoft Word, resulta imprescindible contar con un entorno local que permita visualizar y probar los cambios en tiempo real antes de su despliegue definitivo. Para ello, se suele configurar un servidor web local utilizando tecnologías como Node.js, que facilitan la entrega de los archivos HTML, CSS, JavaScript y recursos estáticos del complemento a través del protocolo HTTPS.

Una característica fundamental de este entorno de desarrollo es el uso obligatorio de conexiones seguras, incluso cuando se trabaja de forma local. Microsoft Office exige que todos los complementos carguen sus recursos mediante HTTPS, lo que obliga a los desarrolladores a emplear certificados SSL autofirmados durante la fase de pruebas. Este requisito responde a razones de seguridad, ya que Word, tanto en su versión de escritorio como en la versión web, bloquea por defecto el contenido que se sirve por HTTP no cifrado.

El comportamiento del servidor se controla mediante scripts definidos en el archivo package.json, que incluyen comandos como *npm start* para iniciar el servidor de desarrollo y levantar el complemento, o *npm stop* para detenerlo. Estos scripts no solo automatizan el arranque del servidor web, sino que también facilitan la carga de certificados, la compilación de los archivos fuente y la apertura de Word en modo de pruebas.

Una vez activo el servidor, es posible realizar lo que se conoce como sideloading, es decir, cargar el complemento en una instancia local de Word sin necesidad de publicarlo en una tienda oficial. Esta técnica resulta especialmente útil para depurar errores, ajustar la interfaz o validar el comportamiento del Add-in en

condiciones reales de uso. Herramientas como Yeoman Generator para Office Add-ins, o el Office Add-in Debugger Extension para Edge, facilitan este proceso y permiten una integración fluida con el entorno de desarrollo.

3.1.5 Uso de la API JavaScript de Office

Uno de los elementos esenciales en el desarrollo de complementos para Microsoft Word es la API JavaScript de Office, conocida como Office.js. Esta biblioteca proporciona un conjunto de herramientas que permite a los desarrolladores interactuar directamente con el contenido del documento, accediendo a su estructura, modificando texto, insertando elementos o capturando datos para su posterior procesamiento. Su diseño está orientado a preservar la seguridad, la compatibilidad y el buen rendimiento dentro del ecosistema Office, ofreciendo una abstracción robusta sobre el modelo interno de los documentos Word.

Para facilitar la programación asíncrona, Office.js adopta un modelo basado en promesas, lo cual permite utilizar las palabras clave *async* y *await* para organizar el flujo de ejecución de manera más clara y eficiente. De esta forma, las operaciones que requieren comunicación con el documento, como leer el cuerpo del texto o insertar contenido en una posición específica, pueden ejecutarse sin bloquear la interfaz de usuario y con un control preciso de los posibles errores.

Una de las funciones más características de esta API es *Word.run*, que encapsula el contexto de ejecución del complemento dentro del documento activo. A través de esta función, se puede acceder a los distintos objetos del modelo de Word, como el cuerpo del documento (*context.document.body*), párrafos concretos, secciones, tablas o controles de contenido. Esta estructura jerárquica permite navegar por el documento de forma ordenada, identificar los elementos relevantes y aplicar las transformaciones deseadas.

Además del acceso al contenido, Office.js expone un objeto global llamado *Office.context*, que proporciona información contextual sobre el entorno de ejecución. Por medio de este objeto es posible obtener, por ejemplo, datos sobre la versión de Word que está utilizando el usuario, el idioma del documento o las capacidades disponibles en la sesión actual. Esta información resulta especialmente útil para adaptar dinámicamente el comportamiento del complemento en función del entorno, garantizando así una mayor compatibilidad y estabilidad.

3.1.6 Resumen de la estructura estándar de un AddIn de Word

Una vez entendido cuáles son los bloques constitutivos de un AddIn la estructura de archivos inicial con la que se puede empezar el desarrollo y que se obtiene con las plantillas de *quick-start* de Microsoft es la siguiente:

```
/mi-word-addin/
├── manifest.xml (o json)
├── src/
│   └── taskpane/
│       ├── taskpane.html
│       ├── taskpane.css
│       └── taskpane.js
├── assets/ (iconos, logos...)
├── package.json (scripts para dev HTTPS)
└── otros (tsconfig.json, README...)
```

Más adelante se profundizará en el aspecto final del AddIn, así como en su estructura una vez integradas las funcionalidades expuestas en la introducción de la memoria.

No obstante, esta estructura dota al proyecto de una integración fluida en Word, así como de seguridad y permisos adecuados que se alinean con los objetivos principales del proyecto: garantizar la seguridad de los datos de los archivos de los profesionales y minimizar la fricción debido al proceso de aprendizaje de uso de la herramienta.

Sistema de Anonimizacion de Datos Sensibles

4.1 Modelos NER

4.1.1 Conceptos fundamentales de los modelos NER

El Reconocimiento de Entidades Nombradas (NER, por sus siglas en inglés Named Entity Recognition) es una técnica fundamental dentro del campo del Procesamiento del Lenguaje Natural (PLN) cuyo objetivo principal consiste en identificar y clasificar automáticamente en un texto aquellas expresiones que hacen referencia a entidades concretas. Tradicionalmente, estas entidades se agrupan en categorías predefinidas como nombres de personas, organizaciones, ubicaciones geográficas, fechas o cantidades monetarias. En esencia, un modelo NER transforma una secuencia de texto sin estructurar en una representación semántica enriquecida, donde ciertos fragmentos del texto son etiquetados como pertenecientes a una clase significativa desde el punto de vista informativo.

Los modelos NER han evolucionado considerablemente desde sus primeras versiones basadas en técnicas estadísticas como los Modelos de Markov Ocultos (HMM) o los Campos Aleatorios Condicionales (CRF)[39], hacia arquitecturas más sofisticadas fundamentadas en redes neuronales profundas. En particular, las arquitecturas basadas en redes BiLSTM con capas de decodificación CRF, y más recientemente los modelos basados en Transformers como BERT o RoBERTa, han demostrado una capacidad sobresaliente para capturar tanto el contexto local como el global de una palabra dentro de una oración, lo cual es crucial para lograr una detección precisa de entidades en textos complejos o ambiguos[50]. Estos modelos aprovechan representaciones distribuidas del lenguaje, conocidas como word embeddings, que codifican el significado de las palabras en función de su uso en grandes corpus textuales, y permiten una generalización mucho más robusta frente a variaciones léxicas o gramaticales.

En el contexto del presente proyecto, se ha adaptado el paradigma de NER para abordar un problema específico: la detección de información sensible en documentos jurídicos, como contratos o informes legales. A diferencia de las tareas convencionales de NER, donde las clases a identificar suelen estar estandarizadas, aquí se propone una redefinición del conjunto de etiquetas con el fin de capturar datos personales y confidenciales que requieren especial protección. Estas nuevas categorías incluyen, entre otras, el número de Documento Nacional de Identidad (DNI), códigos IBAN, direcciones de correo electrónico, nombres completos, direcciones postales o referencias bancarias. La adaptación del modelo a este dominio exige no solo un rediseño del esquema de anotación, sino también una etapa de entrenamiento específica, o fine-tuning, sobre un corpus etiquetado manualmente con estas nuevas clases[15].

4.1.2 Categorización de las Entidades Sensibles

Para poder categorizar numéricamente los datos sensibles que deben ser reconocidos por el modelo NER, el primer paso consistió en definir con precisión qué tipos de información debían considerarse sensibles. En este contexto, cobran especial relevancia las siglas PII (Personally Identifiable Information), término que hace referencia a cualquier dato que permita identificar, rastrear o inferir la identidad de un individuo. Este enfoque no se limita únicamente a nombres propios o identificadores explícitos como el DNI o el número de pasaporte, sino que abarca también información que, de forma directa o indirecta, puede vincularse con una persona. Entre estos datos se incluyen, por ejemplo, el estado civil, las actividades realizadas, la titularidad de propiedades o cualquier otro atributo que pueda contribuir a su individualización.

Con base en este enfoque ampliado, se definió un esquema de 15 etiquetas distintas que representan los tipos de datos sensibles a detectar. Cada una de estas entidades fue codificada numéricamente mediante un sistema de categorización que permite su tratamiento computacional dentro del modelo NER.

- **DNI**: 1
- **NIE**: 2
- **Pasaporte**: 3
- **Teléfono nacional (Nat_PhN)**: 4
- **Teléfono internacional (In_PhN)**: 5
- **Correo electrónico (Email)**: 6

- **IBAN**: 7
- **Referencia catastral (Cat_ref)**: 8
- **Fecha**: 9
- **Valores monetarios**: 10
- **Secuencia numérica (Num_seq)**: 11
- **Nombre propio**: 12
- **Ciudad**: 13
- **País**: 14
- **Calle**: 15

A continuación se muestra como quedarían los ejemplos etiquetados de datos que contienen PII.

4.1.3 Etiquetado de Datos

El etiquetado de los datos se realizó a nivel de token, generando estructuras como los siguientes ejemplos:

```
{
  "tokens": ["El", "DNI", "de", "Juan", "es", "12345678Z", "."],
  "ner_tags": [0, 0, 0, 1, 0, 2, 0]
},
{
  "tokens": ["Contacta", "a", "María", "en", "su", "correo",
  "maria@gmail.com", "."],
  "ner_tags": [0, 0, 12, 0, 0, 0, 6, 0]
}
```

Este formato es completamente compatible con el framework *datasets* de Hugging Face, una biblioteca diseñada para gestionar de manera eficiente grandes volúmenes de datos NLP y preparada para integrarse de forma directa con recursos como Transformers y el objeto Trainer[18]. Al utilizar *load_dataset()* o *Dataset.from_pandas()*, este tipo de estructura se carga en memoria como conjuntos de datos que incorporan columnas de tokens y etiquetas, listas para ser procesadas por los tokenizers y directamente consumidas por el Trainer.

El uso de Trainer permite orquestar todo el ciclo de entrenamiento: procesamiento previo de datos, entrenamiento del modelo, cálculo de métricas (como precisión, recall o F1 con librerías como seqeval) y guardado de puntos de control. Este componente abstrae gran parte de la complejidad del entrenamiento, al mismo tiempo que admite elevados niveles de parametrización, facilitando tareas como fine-tuning de modelos preentrenados en tareas de clasificación secuencial como NER .

En cuanto a la codificación de etiquetas, se opta por una codificación numérica directa (por ejemplo, 0 = "no entidad", 1 = "DNI", 12 = "Nombre propio") en lugar del esquema BIO (Begin–Inside–Outside). Esta elección se basa en dos razones principales. En primer lugar, la codificación numérica simplifica el diseño del esquema de salida, ya que solo requiere una etiqueta por token, evitando la complejidad extra de distinguir clases "begin" e "inside". Por ello, reduce la dimensionalidad del problema al evitar duplicidades en las categorías. En segundo lugar, el enfoque directo resulta más eficiente durante el entrenamiento y la inferencia, pues el modelo únicamente aprende una tarea de clasificación multicategoría por token sin necesidad de gestionar secuencias de prefijos semánticos adicionales .

4.1.4 Generación de Datos sintéticos

Debido a las limitaciones legales y éticas asociadas al uso de datos reales, especialmente en contextos sensibles como el análisis de contratos, se opta por una estrategia de generación de datos sintéticos controlados. La falta de datasets públicos que contengan documentos legales anotados con información personalmente identificable (PII) hace necesario simular este tipo de contenido mediante herramientas avanzadas de generación automática.

Para ello, se emplea el framework LangChain[31], en combinación con modelos de lenguaje de OpenAI, aprovechando su módulo experimental *langchain_experimental.tabular_s*
Este módulo permite definir plantillas estructuradas de entrada, incorporar ejemplos estilo Few-Shot y generar automáticamente textos que replican la estructura, estilo y semántica de contratos reales. Se definieron varias clases base (como ParagraphEPC) que sirvieron de interfaz para agrupar ejemplos sintéticos coherentes, cada uno con alta densidad de entidades etiquetables por el modelo NER.

El proceso de generación sintética se centra en la diversidad y realismo contextual. Se presta especial atención a la variabilidad de formatos, incorporando distintas representaciones válidas de identificadores como DNIs, IBANs, fechas o números de pasaporte. Asimismo, se generan combinaciones contextuales plausibles, como la mención de nombres junto con cargos institucionales, direcciones postales acompañadas de referencias catastrales o cláusulas financieras con valores monetarios en distintos estilos. Para aumentar la cobertura semántica, se generaron ejemplos de fragmentos de contratos de distinta tipología: arrendamientos,

compraventas, acuerdos de financiación, contratos EPC y otros documentos contractuales comunes.

4.1.5 Selección y Justificación del Modelo

Se eligió el modelo xlm-roberta-base por las siguientes razones:

- Multilingüismo: Entrenado en más de 100 idiomas, ideal para textos con terminología internacional.
- Rendimiento probado en NER: Arquitectura Transformer basada en atención y clasificación de tokens.
- Compatibilidad con entrenamiento local: Puede entrenarse en entornos sin necesidad de grandes GPUs.
- Flexibilidad para Fine-Tuning: Permite personalizar las etiquetas del proyecto sin restricciones.

Se descartaron modelos más pesados como xlm-roberta-large por limitaciones computacionales, y modelos monolingües como bert-base-spanish por su menor versatilidad.

4.1.6 Entrenamiento del Modelo

Para ampliar la capacidad de generalización del modelo, el entrenamiento se diseñó bajo una estrategia de Few-Shot Learning, en la que se entrena con muy pocos ejemplos generados de forma sintética previamente (como indicado anteriormente) para inducir al modelo a identificar patrones y relaciones semánticas en las entidades sensibles. Este método, basado en fine-tuning de modelos preentrenados (por ejemplo, BERT o GPT), se ha demostrado eficaz en NER cuando los datos etiquetados son escasos[45]. Los hiperparámetros utilizados para el entrenamiento son los siguientes:

```
    TrainingArguments(
    learning_rate=3e-5,
    num_train_epochs=10,
    per_device_train_batch_size=32,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    gradient_accumulation_steps=2,
    fp16=True
)
```

Ejecutado el entrenamiento, debido a la falta de capacidad computacional se abandona el intento, pese a estar realizando la prueba siguiendo la estrategia de Few-Shot Learning. Un tiempo de espera exagerado, así como un sobrecalentamiento del ordenador en el que se estaba realizando la prueba, concluyen que hay que encontrar un camino alternativo para poder implementar la identificación de datos sensibles.

4.1.7 Conclusiones: Ventajas y Desventajas del enfoque NER

La implementación de modelos de Reconocimiento de Entidades Nombradas (NER) constituye un componente clave en el desarrollo de sistemas automatizados de detección y anonimización de información sensible. Su capacidad para identificar entidades a nivel de token, teniendo en cuenta el contexto lingüístico y semántico, ha demostrado ser especialmente eficaz en documentos contractuales y jurídicos, donde la precisión es fundamental.

Ventajas del enfoque NER:

- Alta precisión en clasificación token a token, gracias a la capacidad de los modelos para contextualizar cada unidad léxica.
- Adaptabilidad a nuevas clases de entidad, mediante procesos de fine-tuning con ejemplos específicos del dominio.
- Escalabilidad multilingüe, al aprovechar modelos preentrenados que permiten aplicarse en distintos idiomas sin rediseño estructural.

Desventajas y limitaciones:

- Elevada demanda computacional, especialmente en fase de entrenamiento, lo que dificulta su ejecución en entornos sin GPU o con recursos limitados.
- Dependencia de datos etiquetados de calidad, sin los cuales el modelo puede generalizar de forma deficiente o aprender patrones erróneos.

4.2 Evaluación de modelos de IA locales

El siguiente paso, con el objetivo de superar la barrera de la capacidad computacional impuesta por los modelos NER, consiste en evaluar el rendimiento de LLMs de tamaño medio y pequeño desplegados localmente. Este enfoque busca simular un entorno de tipo API, en el que se envían párrafos a anonimizar y se recibe como respuesta un JSON con los datos sensibles detectados. En caso de obtener buenos

resultados en la detección, esta estrategia presentaría ventajas significativas, como la reducción de costes asociados al uso de servicios en la nube o APIs externas, así como la posibilidad de operar en entornos offline.

4.2.1 Selección del modelo: LLAMA3.2 3B

Para encontrar la forma más sencilla y eficiente de desplegar un modelo de forma interna, se llegó a la conclusión que se ha de emplear Ollama[40], una plataforma que permite ejecutar modelos de lenguaje de gran tamaño (LLM) de forma local, simplificando tanto su despliegue como su integración en distintos entornos de desarrollo. Esta herramienta, impulsada por Simon Willison y su equipo, está diseñada para facilitar el uso de modelos de inteligencia artificial mediante una interfaz de línea de comandos intuitiva y una arquitectura que gestiona automáticamente la descarga, ejecución y actualización de modelos. Ollama destaca por su eficiencia, su compatibilidad con arquitecturas ligeras, incluidas aquellas que no requieren GPU de alto rendimiento, y por ofrecer una API local que permite una integración sencilla con otros sistemas.

Dentro de este entorno, se opta por utilizar el modelo LLAMA3.2 de 3B de parámetros, una versión ligera basada en la arquitectura LLaMA3, por su adecuado equilibrio entre precisión y consumo de recursos. Este modelo es elegido porque según los benchmarks[12] disponibles en el momento de la selección aparece como uno de los más versátiles y con mejor rendimiento para su tamaño, siendo en teoría capaz de llevar a cabo tareas básicas de procesamiento del lenguaje natural, como la detección de patrones de texto o la extracción de información estructurada, y resulta lo suficientemente eficiente como para ejecutarse en equipos convencionales sin necesidad de infraestructura especializada.

4.2.2 Instalación del modelo en local

Para poder ejecutar el modelo en local primero se ha de instalar la plataforma directamente a partir de su página web[40]. Una vez instalada la herramienta el siguiente paso consiste en descargar el modelo deseado (Llama3.2). Tras ser descargado el modelo podemos realizar unas pruebas dentr de la consola de windows que nos permiten ver y entender el funcionamiento del modelo mediante el comando *ollama run llama3.2*:

```
Prompt (en español):
Hola, ¿cómo estás? Quiero que me indiques tu principal fortaleza y debilidad.
Solamente quiero que me respondas con una cosa para cada caso.

Respuesta:
```

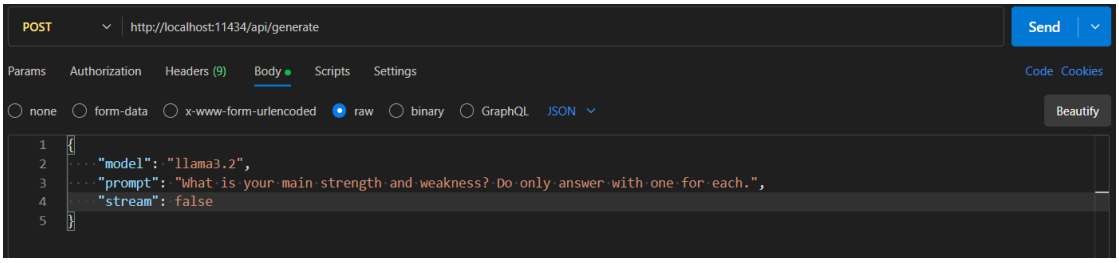


Figure 4.1: Realiza llamada a API local de Ollama para poner a prueba modelo Llama3.2

```
Fortaleza: Mi capacidad para entender y responder a preguntas de manera rápida
y precisa,lo que la hace ideal para tareas como soporte al cliente o
traducción de idiomas.

Debilidad: No tengo una comprensión real de la emoción humana,
lo que puede llevarme a no poder proporcionar apoyo emocional adecuado en
ciertas situaciones.
```

Al ejecutar el comando mencionado anteriormente (*ollama run llama3.2*), ollama se ejecuta como un servicio local en el puerto 11423 por defecto que permite simular el comportamiento de llamadas a API en las aplicaciones. Para poner esto a prueba, se hace uso de la aplicación Postman6.7:

```
{
"model": "llama3.2",
"created_at": "2025-06-29T08:45:27.0719932Z",
"response": "I can provide information and entertainment, but I can't
currently take actions on your behalf. For example, I can plan a custom
travel itinerary, but I can't buy tickets or book hotels. I can write you
an email, but I can't send it. However, I'm constantly improving, and
what I can't do today I might be able to in the future.",
"done": true,
"done_reason": "stop",
"context": [
    128006,
    9125,
    ...
],
"total_duration": 2308026400,
"load_duration": 42221400,
"prompt_eval_count": 41,
```

Campo	Explicación
`model`	Nombre del modelo usado. En este caso: `llama3.2`.
`created_at`	Fecha y hora (en formato ISO 8601) de la generación de la respuesta.
`response`	Texto generado por el modelo como respuesta al `prompt`.
`done`	Booleano que indica si el modelo terminó de generar la respuesta (`true` = completado).
`done_reason`	Razón por la que el modelo detuvo la generación. Generalmente `stop` (fin normal) o `length` (límite de tokens).
`context`	Lista de tokens (enteros) que representan el estado interno tras procesar el prompt. Útil para continuar la conversación o mantener el contexto.
`total_duration`	Tiempo total (en nanosegundos) que tardó toda la operación: carga + evaluación del prompt + generación.
`load_duration`	Tiempo (en ns) que tardó en cargar el modelo si no estaba ya cargado en memoria. Puede ser cero si ya estaba listo.
`eval_count`	Número de tokens evaluados del prompt original (entrada).
`eval_duration`	Tiempo en ns que tardó el modelo en analizar (tokenizar y evaluar) el prompt.
`eval_count`	Número de tokens generados como respuesta (`response`).
`eval_duration`	Tiempo que tardó en generar la respuesta completa (en ns).

Table 4.1: Descripción de los campos devueltos por el endpoint de Ollama

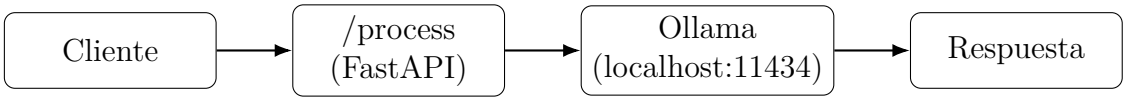
```
    "prompt_eval_duration": 10000000,
    "eval_count": 77,
    "eval_duration": 2253000000
}
```

La explicación del significado de cada elemento de la siguiente respuesta se puede entender mediante la siguiente tabla 4.1.

4.2.3 Integración con backend FastAPI y Funcionamiento del sistema

Para interactuar con Ollama desde una aplicación web, se desarrolla un backend en FastAPI. Este backend expone un endpoint */process* que recibe un listado de

párrafos, los procesa mediante el modelo Ollama y devuelve las respuestas. El esquema general que sigue el funcionamiento es el siguiente:



A continuación el fragmento de código que orquesta las solicitudes:

```python
import requests

class OllamaService:
    def __init__(self, model_name: str = "llama3:3b"):
        self.model_name = model_name
        self.api_base = "http://localhost:11434"

    def process_text(self, text: str) -> str:
        try:
            response = requests.post(
                f"{self.api_base}/api/generate",
                json={
                    "model": self.model_name,
                    "prompt": text,
                    "stream": False
                }
            )
            response.raise_for_status()
            return response.json().get("response", "")
        except requests.exceptions.RequestException as e:
            print(f"Error al conectarse con Ollama: {e}")
            return "Error en la generacion de texto."
```

Listing 4.1: Servicio de integración local con Ollama en Python

4.2.4 Conclusiones: Ventajas y desventajas del enfoque local

Una vez implementada la arquitectura y completadas las pruebas de identificación de datos sensibles, utilizando ejemplos generados sintéticamente para el entrenamiento del modelo NER, emergen rápidamente las primeras limitaciones de los modelos pequeños. En primer lugar, no siguen con precisión las instrucciones proporcionadas en los prompts. Además, muestran dificultades para generar respuestas consistentes que detecten correctamente la información sensible y que, al mismo tiempo, sean fácilmente procesables dentro del marco general de la aplicación web. Es común que produzcan respuestas con sobreinformación o incluso

contenido alucinado, desviándose de las instrucciones y comentando aspectos irrelevantes del ejemplo analizado. Para identificar si era una cuestión del modelo o del tamaño se llevan a cabo pruebas con otros modelos, como por ejemplo *deepseek-r1* de 1.5B de parámetros obteniendo el mismo resultado insuficiente. Por tanto, se concluye que, aunque el despliegue local de modelos de lenguaje opensource de tamaño reducido ofrece ventajas notables en términos de privacidad total de los datos, bajo coste operativo, y mayor control y flexibilidad en el diseño, estos modelos no resultan adecuados para tareas de categorización que deban integrarse en arquitecturas más complejas, donde se exige un tratamiento más riguroso y preciso de la información.

4.3 Enfoque híbrido basado en patrones (regex y bases de datos)

La tercera fase del sistema de detección y anonimización se fundamenta en un enfoque híbrido que combina expresiones regulares (regex) con búsquedas controladas en bases de datos optimizadas de nombres y apellidos. Esta aproximación tiene como objetivo incrementar la precisión y cobertura del sistema, permitiendo la identificación tanto de patrones estructurados como de entidades nombradas que podrían representar datos sensibles en documentos textuales.

4.3.1 Diseño de la arquitectura

Desde el punto de vista arquitectónico, esta fase está implementada en un servidor backend desarrollado con el microframework Flask. Las rutas del sistema están organizadas en un blueprint bajo el prefijo */api*, donde se define, entre otros, el endpoint */api/findsensibleinfo/es*. Este punto de entrada recibe peticiones POST que contienen una lista de párrafos en español, y devuelve los fragmentos identificados como sensibles tras un análisis exhaustivo.

El proceso de detección se articula en varios niveles. En primer lugar, se aplican patrones predefinidos mediante expresiones regulares, diseñados para capturar estructuras como documentos de identidad (DNI, NIE, pasaportes), números telefónicos, códigos IBAN, fechas en distintos formatos y valores monetarios. Estos patrones son evaluados sobre cada párrafo del texto de entrada, extrayendo coincidencias explícitas mediante la función *re.findall*.

Complementariamente, se emplean funciones heurísticas específicas contenidas en los módulos *funciones.py* y *functions.py*, las cuales permiten capturar otros tipos de entidades sensibles. Entre ellas se encuentran funciones para identificar palabras inusualmente largas (posibles identificadores únicos), direcciones de correo

electrónico (detectadas por la presencia del carácter '@') o secuencias próximas a palabras clave como "DNI", "ID", "NIF", entre otras.

Uno de los componentes más relevantes de esta fase es la verificación cruzada contra bases de datos de nombres y apellidos tanto en español como en inglés. Para ello, se construyen índices optimizados a partir de los archivos *esp_nombres.json*, *esp_apellidos.json*, *uk_names.json* y *uk_surnames.json*. Estos índices agrupan los nombres por letra inicial, permitiendo reducir la complejidad de búsqueda. Durante el análisis del texto, se realiza una normalización que elimina tildes y convierte el texto a minúsculas. Luego, se compara cada palabra con el índice correspondiente a su inicial, permitiendo así detectar coincidencias precisas con nombres y apellidos comunes.

Una vez detectados todos los elementos sensibles, los resultados se agrupan, eliminando duplicados para evitar redundancias, y se devuelven al cliente en una única respuesta estructurada. Este enfoque híbrido permite combinar la eficiencia de los patrones formales con la flexibilidad del conocimiento semántico encapsulado en las bases de datos, logrando una detección más robusta y adaptable a distintos contextos lingüísticos.

4.3.2 Conclusiones: ventajas y desventajas del enfoque híbrido

Este enfoque representa el primer paso efectivo hacia la detección fiable de datos sensibles, arrojando resultados correctos y consistentes en un alto número de casos. La verificación cruzada con bases de datos optimizadas de nombres y apellidos, tanto en español como en inglés, demuestra una robustez destacable, y tras múltiples iteraciones en el ajuste de los patrones mediante expresiones regulares, se ha logrado alcanzar una cobertura aproximada del 80 por ciento de los datos sensibles presentes en los textos evaluados. Esta cifra, que supone una mejora significativa respecto a los métodos anteriores, no obstante, sigue siendo insuficiente para aspirar a una solución completamente fiable y, por tanto, comercialmente viable.

Uno de los principales desafíos de este enfoque reside en la alta tasa de falsos negativos. Cuando el sistema no es capaz de identificar determinadas entidades sensibles, obliga al profesional a realizar una revisión manual del documento completo, neutralizando así la ganancia de eficiencia que se pretendía obtener. En este sentido, la falta de comprensión contextual limita la aplicabilidad del sistema a entornos reales, donde la precisión en la detección debe ser prácticamente total.

El obstáculo más relevante se presenta cuando el sistema se enfrenta a información ambigua desde el punto de vista léxico, cuya sensibilidad depende exclusivamente del contexto en el que aparece. Un ejemplo ilustrativo es el caso de entidades jurídicas con nombres compuestos por términos comunes, como *Valle Plateado S.L.*. Aisladas, las palabras "valle" o "plateado" no permiten inferir datos personales; sin embargo, dentro de un fragmento en el que se especifican

las partes de un contrato, dicha denominación corresponde inequívocamente a una entidad concreta y debe, por tanto, ser tratada como información sensible. Este tipo de ambigüedad semántica, habitual en documentos legales, escapa al alcance de los enfoques basados únicamente en patrones o coincidencias léxicas.

En definitiva, si bien este enfoque híbrido consigue buenos resultados en escenarios directos y poco ambiguos, su incapacidad para incorporar el contexto semántico de las palabras limita su eficacia global. El avance es notorio respecto a las soluciones anteriores, pero la necesidad de integrar mecanismos de comprensión contextual más sofisticados resulta imprescindible para alcanzar una solución verdaderamente fiable y automatizable en el marco profesional.

4.4 Despliegue del sistema de anonimización en AWS

Como siguiente paso, y con el objetivo de incorporar el contexto de los datos en su evaluación y extracción, se lleva a cabo una investigación sobre la viabilidad de utilizar modelos de lenguaje de mayor tamaño que los empleados en la fase anterior. Esta investigación conduce a la elección de AWS Bedrock como solución.

4.4.1 Introducción al uso de AWS en el proyecto

Amazon Web Services es actualmente uno de los proveedores de servicios en la nube más consolidados del mercado, con una cuota superior al 30% en la industria global de infraestructura como servicio (IaaS), según datos de Gartner (2023). AWS ofrece un amplio abanico de herramientas y servicios gestionados que permiten abstraerse de la infraestructura física, reducir el tiempo de desarrollo, y garantizar tanto la escalabilidad como la resiliencia de las aplicaciones desplegadas.

Uno de los retos principales en el tratamiento de datos sensibles es garantizar su confidencialidad durante todas las etapas de procesamiento, especialmente cuando se utilizan modelos LLM. En este sentido, el ecosistema de AWS aporta una serie de ventajas diferenciales[10]: la posibilidad de seleccionar regiones específicas para el procesamiento, en este caso, Europa Central (eu-central-1), el cumplimiento de normativas como el Reglamento General de Protección de Datos (RGPD)[24], y una infraestructura segura certificada bajo estándares internacionales como ISO/IEC 27001[8], SOC 2 y CIS AWS Foundations Benchmark.

La elección de una arquitectura basada en funciones Lambda (modelo serverless), junto con servicios como API Gateway y AWS Bedrock, ha permitido construir un sistema modular, flexible y de bajo mantenimiento. Esta combinación de tecnologías proporciona una capa de orquestación que permite, por un lado, recibir

y procesar documentos a través de endpoints HTTP y, por otro, invocar dinámicamente modelos de lenguaje que se ejecutan en entornos gestionados, garantizando así tanto el rendimiento como la protección de los datos durante su análisis.

4.4.2 AWS Bedrock: arquitectura y garantías de privacidad

AWS Bedrock[3] es un servicio gestionado dentro del ecosistema de Amazon Web Services que permite a desarrolladores y organizaciones acceder a modelos fundacionales de lenguaje de última generación a través de una API, sin necesidad de desplegar, entrenar ni mantener infraestructura propia de aprendizaje automático. Esta propuesta se enmarca dentro del modelo "Foundation Models as a Service", donde los modelos, alojados y mantenidos por los proveedores originales (como Mistral, Anthropic o AI21), son ofrecidos directamente desde la plataforma de AWS como servicios invocables, eliminando así barreras técnicas y operativas para su integración en entornos productivos.

El objetivo principal de AWS Bedrock es democratizar el acceso a modelos LLM robustos y versátiles, permitiendo a los equipos de desarrollo centrarse en la lógica de negocio y la seguridad del sistema sin tener que gestionar complejos pipelines de entrenamiento o ajustar entornos de alto rendimiento. Desde el punto de vista arquitectónico, Bedrock abstrae completamente la infraestructura subyacente, garantizando a su vez que las llamadas al modelo se procesan de manera segura, escalable y conforme a las normativas de protección de datos vigentes.

Una de las características más relevantes de AWS Bedrock en el contexto de este Trabajo de Fin de Grado es su compromiso explícito con la privacidad de los datos. Según la documentación oficial del servicio[5], Bedrock no retiene ni almacena los datos procesados por los modelos, y no los utiliza para reentrenar modelos fundacionales. Todo el procesamiento se realiza en tiempo real, y los datos son descartados inmediatamente tras completar la inferencia. Este enfoque cumple con el principio de minimización de datos establecido en el artículo 5 del Reglamento General de Protección de Datos (RGPD), y proporciona garantías adicionales frente a otros proveedores en los que el tratamiento puede implicar almacenamiento transitorio o reutilización de datos para mejorar el modelo.

4.4.3 Funcionamiento del modelo de detección y anonimización

Todos los archivos de detección de PII están presentes en el repositorio de github ubicado en el Anexo B 10. El sistema de detección y anonimización de datos sensibles desarrollado en esta fase se fundamenta en un enfoque modular por fases,

que permite dividir y clasificar de manera más precisa los distintos tipos de información que pueden considerarse protegidos según el marco normativo europeo. En lugar de utilizar una única llamada monolítica a un modelo de lenguaje que intente resolver todas las tareas en simultáneo, se optó por una arquitectura de doble pasada (pipeline bifásico) que separa el procesamiento en dos bloques lógicos: detección de datos personales e identificadores, y detección de datos contextuales y referencias indirectas.

En la fase 1, el sistema identifica expresamente información personal directa, como nombres y apellidos de personas físicas, números de teléfono, correos electrónicos, documentos nacionales de identidad (DNI y NIE), identificadores fiscales (CIF/NIF), números de pasaporte, cuentas bancarias (IBAN) y otros patrones numéricos sensibles. Estos elementos son los más directamente vinculables a un individuo y su presencia en documentos jurídicos suele requerir medidas estrictas de anonimización según el principio de confidencialidad reforzada del RGPD.

```
{
"Nombres": ["<nombre1>", "<nombre2>"],
"Apellidos": ["<apellido1>", "<apellido2>"],
"DNI": ["<dni1>", "<dni2>"],
"NIE": ["<nie1>", "<nie2>"],
"Pasaporte": ["<pasaporte1>", "<pasaporte2>"],
"Numero_telefonico": ["<numero1>", "<numero2>"],
"IBAN": ["<iban1>", "<iban2>"],
"Emails": ["<email1>", "<email2>"],
"Identificador_fiscal": ["<identificador1>", "<identificador2>"],
"Otros_Identificadores_Numericos": ["<otro_identificador1>", "<otro_identificador
}
```

En la fase 2, el sistema se centra en elementos de carácter contextual que, aunque no constituyen datos personales en sentido estricto, pueden servir para inferir identidades o situaciones sensibles en combinación con otros. Entre estos se encuentran referencias catastrales, direcciones, nombres de empresas, fechas clave, cantidades monetarias, porcentajes, localidades, países, hojas registrales, entre otros. La capacidad de identificar este tipo de información resulta fundamental en sectores como el legal, donde las implicaciones de privacidad no se limitan únicamente a datos identificativos, sino también a patrones de contexto.

```
{
"Referencia_catastral": ["<referencia1>", "<referencia2>"],
"Fechas": ["<fecha1>", "<fecha2>"],
"Valores_monetarios": ["<valor1>", "<valor2>"],
"Regiones": ["<region1>", "<region2>"],
```

```
"Nombres_Empresas": ["<empresa1>", "<empresa2>"],
"Paises": ["<pais1>", "<pais2>"],
"Ciudades": ["<ciudad1>", "<ciudad2>"],
"Direcciones": ["<direccion1>", "<direccion2>"],
"Porcentajes": ["<porcentaje1>", "<porcentaje2>"],
"Cuantias_en_palabras": ["<cuantia_en_palabra1>", "<cuantia_en_palabra2>"],
"Codigos_postales": ["<codigo_postal1>", "<codigo_postal2>"],
"Numero_participaciones": ["<participacion1>", "<participacion2>"],
"Hojas_registrales": ["<hoja_registral1>", "<hoja_registral2>"],
"Otros": ["<otros1>", "<otros2>"]
}
```

El pipeline general de ejecución sigue una estructura claramente definida, optimizada para maximizar el rendimiento del sistema sin comprometer la precisión en la detección. El primer paso del proceso consiste en recibir como entrada un documento completo, normalmente en formato texto plano. A continuación, este documento se divide en párrafos utilizando saltos de línea como delimitadores lógicos. La función *split_into_paragraphs()* se encarga de realizar esta segmentación inicial limpiando espacios en blanco innecesarios.

Una vez obtenida la lista de párrafos, estos se agrupan en "chunks" o fragmentos concatenados de tamaño controlado (por ejemplo, 25 párrafos por chunk). Esta estrategia responde a la necesidad de mantener el contexto local dentro de cada envío al modelo, sin sobrepasar los límites de tokens permitidos por el modelo fundacional en AWS Bedrock. La función *create_chunks()* es la encargada de esta tarea.

Cada uno de estos chunks es procesado de forma concurrente mediante un sistema de hilos paralelos que distribuye las tareas de inferencia entre distintas instancias de ejecución. Para ello se utiliza *ThreadPoolExecutor*[9] con un número máximo de hilos definido por la constante *MAX_WORKERS*. Esto permite escalar horizontalmente el procesamiento y reducir drásticamente la latencia total, un aspecto crítico para aplicaciones en producción que requieren tiempos de respuesta razonables.

Cada chunk es enviado secuencialmente a ambas fases del pipeline. En la primera, se utiliza la función *process_chunk_fase1()*, mientras que en la segunda se ejecuta *process_chunk_fase2()*. Ambas funciones preparan un prompt específico para cada modelo, siguiendo una plantilla predefinida con formato claro, objetivo y determinista. Para construir estos prompts se hace uso de la librería Jinja2[43], que permite insertar el contenido del chunk dentro de un marco estático bien definido, garantizando así consistencia en las respuestas del modelo. Este mecanismo se refleja en las *funciones formateo_call_datos_personales()* y *formateo_call_datos_contextuales()*.

```
    template_string = """
    <...>
    <data>{{ transcript }}</data>
    <...>
    """
    template = Template(template_string)
    prompt = template.render(transcript=transcript)
```

Listing 4.2: Generación del prompt para datos personales

Finalmente, una vez procesadas ambas fases, los resultados son combinados en una única estructura mediante la función *combine_results_two_phases()*. Esta etapa elimina duplicados, preserva el orden de aparición original y clasifica los datos detectados según su categoría. El resultado final es un objeto JSON estructurado que recoge toda la información sensible encontrada en el documento, facilitando su posterior anonimización o revisión manual.

```
for key in create_empty_result_template_fase1().keys():
    for result in results_fase1:
        for item in result.get(key, []):
            if item not in combined[key]:
                combined[key].append(item)
```

Listing 4.3: Combinación de resultados detectados

Este enfoque por fases, combinado con técnicas de paralelización y estandarización de prompts, ha demostrado ser eficaz tanto desde el punto de vista computacional como en términos de calidad de detección, ofreciendo un equilibrio robusto entre coste, rendimiento y precisión.

4.4.4 Procesamiento interno en AWS Lambda

El componente central del sistema de detección y anonimización de datos sensibles está desplegado en AWS Lambda[6], un servicio de computación sin servidor (*serverless*) que permite ejecutar funciones en la nube sin necesidad de gestionar servidores, escalar infraestructura ni configurar entornos persistentes. En esencia, AWS Lambda ejecuta código en respuesta a eventos, como peticiones HTTP o llamadas internas, y solo cobra por el tiempo de cómputo consumido, lo cual lo convierte en una solución altamente eficiente y económica para cargas intermitentes o por demanda.

En el contexto de este proyecto, AWS Lambda ha sido la elección natural por varias razones. En primer lugar, su naturaleza event-driven se adapta perfectamente al modelo de servicio de detección bajo demanda: cada vez que se recibe un documento, este se procesa en una función aislada, garantizando así independencia entre ejecuciones, escalabilidad automática y tolerancia a fallos. En segundo lugar, Lambda proporciona una integración fluida con otros servicios de AWS como API

Gateway, Bedrock, CloudWatch y IAM, lo que permite construir una arquitectura modular y segura, con permisos restringidos por función.

Una de las limitaciones inherentes a AWS Lambda es el tamaño reducido del entorno de ejecución por defecto (50MB comprimidos), lo que complica la incorporación de bibliotecas externas no incluidas en el entorno base de Python. Para solventar esta limitación, se ha hecho uso de una funcionalidad específica del servicio: los Lambda Layers[11]. Los layers permiten adjuntar conjuntos de dependencias adicionales que se cargan junto al código principal en tiempo de ejecución. En este caso, se han utilizado layers personalizados que incluyen librerías como:

- Jinja2, utilizada para generar los prompts que se envían al modelo de lenguaje de manera estandarizada.
- boto3[7], la SDK oficial de AWS en Python, necesaria para invocar la API de Bedrock.
- re, el módulo de expresiones regulares, para el análisis posterior de la salida textual del modelo;

La arquitectura interna de la función Lambda está organizada de forma modular para garantizar claridad y extensibilidad del código. La función principal, denominada *lambda_handler*, actúa como punto de entrada para todas las peticiones. Su lógica se basa en deserializar el cuerpo de la petición HTTP entrante, identificar el tipo de función solicitada por el cliente (por ejemplo, detección de datos o interacción conversacional) y redirigir el control al bloque funcional correspondiente.

En concreto, se definen dos subfunciones principales: *handle_sensitive_data_detection()*, encargada de procesar los documentos mediante el pipeline de doble fase explicado anteriormente; y *handle_claudia_chat()*, que permite utilizar el mismo modelo para generar respuestas conversacionales en un contexto asistencial. Esta separación de responsabilidades permite mantener el código limpio y escalable, favoreciendo además la incorporación de futuras funcionalidades dentro del mismo backend.

Este diseño modular no solo mejora la mantenibilidad del sistema, sino que también permite aprovechar los mecanismos de paralelización interna y la naturaleza sin estado de las funciones Lambda. Cada ejecución parte de cero, lo que garantiza que no se acumulen datos ni estado entre invocaciones, un factor fundamental para cumplir con las exigencias de privacidad en el tratamiento de datos sensibles.

4.4.5 Paralelización del procesamiento y división en chunks

Una de las principales dificultades a la hora de procesar documentos extensos con modelos de lenguaje es su límite intrínseco en el número de tokens o caracteres que pueden analizar por cada inferencia. Los modelos disponibles en AWS Bedrock, como el utilizado en este proyecto (Pixtral Large, basado en la arquitectura de Mistral), presentan un límite de contexto de entre 8.000 y 32.000 tokens, lo que obliga a dividir previamente el documento de entrada para permitir su procesamiento completo sin pérdida de información o errores por desbordamiento. Para abordar este desafío, se establece desde el propio Add-in de Word un control sobre cómo se envían los párrafos a la función Lambda, evitando así tareas adicionales de segmentación (chunking) y tratamiento de espacios, saltos de línea u otros elementos de formato. Aunque más adelante se detallará en profundidad el funcionamiento del Add-in, en esta fase se opta por respetar la estructura de párrafos definida por el editor de Word, en lugar de enviar el texto completo y subdividirlo activamente. Esta decisión busca preservar la coherencia del contenido y facilitar un procesamiento más preciso. Una vez obtenidos los párrafos individuales, estos se agrupan en bloques de tamaño controlado a través de la función *create_chunks()*. Cada chunk consiste en la concatenación de un número determinado de párrafos, establecido en el parámetro *CHUNK_SIZE*, con el fin de generar fragmentos suficientemente grandes para mantener el contexto local, pero suficientemente pequeños para no superar el límite de tokens del modelo de lenguaje. En el sistema implementado, este valor se ha fijado empíricamente en 25 párrafos, tras evaluar el equilibrio entre calidad del análisis contextual y velocidad de inferencia. El siguiente paso en el pipeline es el procesamiento en paralelo de estos chunks. Para minimizar la latencia y reducir el tiempo total de ejecución del sistema, se emplea el módulo *concurrent.futures*[44] de Python, concretamente el componente *ThreadPoolExecutor*. Este permite distribuir las tareas de inferencia entre varios hilos de ejecución simultáneos, aprovechando al máximo las capacidades de concurrencia dentro del entorno de ejecución de AWS Lambda.

Cada chunk es procesado de forma independiente en dos fases consecutivas: detección de datos personales (*process_chunk_fase1*) y detección de datos contextuales (*process_chunk_fase2*). Estas funciones son invocadas en paralelo sobre los distintos bloques mediante *executor.submit*, lo que permite lanzar múltiples llamadas a AWS Bedrock de forma concurrente y recolectar los resultados conforme se van completando. El valor del parámetro *MAX_WORKERS*, que define el número máximo de hilos concurrentes, se ha establecido en 15 tras pruebas iterativas que revelaron que este era el punto óptimo para minimizar los tiempos sin saturar la capacidad de ejecución de Lambda ni generar errores de timeout o throttling por parte de la API de Bedrock.

```
1 # Fase 1: Procesar datos personales
```

```
with concurrent.futures.ThreadPoolExecutor(max_workers=effective_workers) as executor:
    future_to_chunk = {executor.submit(process_chunk_fase1, chunk): i for i, chunk in enumerate(chunks)}

    for future in concurrent.futures.as_completed(future_to_chunk):
        chunk_index = future_to_chunk[future]
        try:
            result = future.result()
            print(f"Fase 1: Chunk {chunk_index + 1}/{len(chunks)} procesado exitosamente")
            results_fase1.append(result)
        except Exception as e:
            print(f"Error en fase 1, chunk {chunk_index + 1}/{len(chunks)}: {str(e)}")
            results_fase1.append(create_empty_result_template_fase1())

print(f"Iniciando procesamiento paralelo fase 2 con {effective_workers} trabajadores")

# Fase 2: Procesar datos contextuales
with concurrent.futures.ThreadPoolExecutor(max_workers=effective_workers) as executor:
    future_to_chunk = {executor.submit(process_chunk_fase2, chunk): i for i, chunk in enumerate(chunks)}

    for future in concurrent.futures.as_completed(future_to_chunk):
        chunk_index = future_to_chunk[future]
        try:
            result = future.result()
            print(f"Fase 2: Chunk {chunk_index + 1}/{len(chunks)} procesado exitosamente")
            results_fase2.append(result)
        except Exception as e:
            print(f"Error en fase 2, chunk {chunk_index + 1}/{len(chunks)}: {str(e)}")
            results_fase2.append(create_empty_result_template_fase2())
```

Listing 4.4: Procesamiento concurrente en ambas fases

La elección adecuada de *CHUNK_SIZE* y *MAX_WORKERS* es fundamental para el rendimiento global del sistema. Si los chunks son demasiado pequeños, el número total de llamadas al modelo aumenta considerablemente, incrementando los costes operativos y la carga computacional. Si, por el contrario, son demasiado grandes, se corre el riesgo de superar el límite de contexto del modelo o de que se pierda

precisión en la segmentación semántica. De forma análoga, un valor demasiado alto de *MAX_WORKERS* puede saturar los límites de concurrencia de AWS Lambda o provocar una penalización en la latencia si las tareas se ejecutan en cola.

En conjunto, esta arquitectura basada en fragmentación semántica y paralelización concurrente permite procesar documentos extensos de forma eficiente, respetando las limitaciones técnicas del modelo y maximizando el rendimiento dentro de los márgenes económicos y temporales establecidos para un sistema de análisis legal en producción.

4.4.6 Integración de la función Lambda con Api Gateway

Para permitir que clientes externos puedan interactuar con el sistema de detección y anonimización desarrollado, ha sido necesario exponer su funcionalidad a través de una interfaz accesible mediante solicitudes HTTP. Este propósito se ha cumplido mediante la integración de AWS API Gateway[2], un servicio completamente gestionado que permite definir, desplegar y mantener interfaces de programación de aplicaciones (APIs) REST o HTTP de forma segura, escalable y sin necesidad de servidores propios.

API Gateway actúa como punto de entrada para las solicitudes que se deseen dirigir a la función Lambda desplegada. En este proyecto, su rol ha sido clave como interfaz pública de comunicación, permitiendo encapsular la complejidad del backend, restringir el acceso mediante políticas de control, y aplicar transformaciones o validaciones previas a la ejecución del código. Al conectarse directamente con funciones Lambda, API Gateway permite construir arquitecturas completamente sin servidor (serverless), donde no es necesario mantener servidores web dedicados para recibir y enrutar solicitudes.

En el sistema desarrollado se han definido dos endpoints principales en API Gateway:

- Un endpoint general que recibe solicitudes de tipo POST para la función de detección de datos sensibles, el cual acepta como cuerpo (body) un objeto JSON con los campos "function": "detectSensitiveData" y "texto", correspondiente al contenido del documento a analizar.

- Un segundo endpoint, también de tipo POST, que permite invocar al modelo en modo conversacional a través del parámetro "function": "claudia", habilitando una interfaz estilo chatbot como asistencia lingüística avanzada.

Ambos endpoints están configurados para enrutar directamente hacia la misma función Lambda (*lambda_handler*), que luego determina internamente a qué subfunción delegar la lógica de negocio, como se explicó en apartados anteriores. Esta

estrategia permite mantener una única función Lambda con múltiples comportamientos, reduciendo los costes de mantenimiento y facilitando la escalabilidad lógica.

En cuanto a la compatibilidad con aplicaciones web front-end o clientes en navegadores, un aspecto fundamental de la configuración ha sido el correcto manejo de CORS (Cross-Origin Resource Sharing). Dado que el frontend del sistema puede estar desplegado en un dominio distinto al de la API, es necesario permitir explícitamente la comunicación entre ambos. Para ello, API Gateway ha sido configurado para incluir los encabezados CORS adecuados en las respuestas, en particular:

- Access-Control-Allow-Origin: *, que permite peticiones desde cualquier origen (o desde un dominio específico en producción).
- Access-Control-Allow-Methods: POST, OPTIONS, habilitando los métodos requeridos por la lógica de negocio y preflight.
- Access-Control-Allow-Headers: Content-Type, para aceptar solicitudes con datos en formato JSON.

Por otro lado, también se han añadido cabeceras de seguridad en las respuestas, como Content-Type: application/json; charset=utf-8, para asegurar la correcta interpretación de los datos por parte del cliente y prevenir errores de codificación en respuestas que incluyen acentos o caracteres especiales. Esta integración entre Lambda y API Gateway permite disponer de una API robusta, segura y flexible, capaz de recibir peticiones JSON, desencadenar el procesamiento de datos en segundo plano y devolver resultados anonimizados en cuestión de segundos. Todo ello sin necesidad de gestionar servidores físicos ni plataformas intermedias, lo que constituye una ventaja competitiva clara en entornos LegalTech y sistemas SaaS distribuidos.

4.4.7 Costes asociados y latencia del sistema

Uno de los aspectos más relevantes en la evaluación de soluciones en la nube es el equilibrio entre rendimiento y coste. En el sistema desarrollado, tanto la función de procesamiento como la inferencia mediante modelos LLM están desplegadas sobre servicios gestionados de AWS, lo cual permite analizar con precisión los costes por unidad de ejecución, así como las implicaciones en términos de latencia.

En primer lugar, la ejecución de la lógica principal se realiza mediante AWS Lambda, un servicio de facturación por uso basado en dos métricas: el número de invocaciones y el tiempo de ejecución en milisegundos, ponderado por la cantidad de memoria asignada. Para este proyecto, se ha utilizado una configuración de

1.024MB de memoria, con una duración promedio por ejecución de entre 6 y 12 segundos por documento, dependiendo del número de chunks. Dado que Lambda incluye un tramo gratuito mensual de un millón de invocaciones y 400.000 GB-segundos, su coste en entornos de desarrollo o bajo volumen es prácticamente nulo. Sin embargo, en producción, su precio base es de aproximadamente 0,00001667 USD por GB-segundo, lo que representa un coste medio por documento de entre 0,01 y 0,03 USD, dependiendo del tamaño.

En cuanto al coste del procesamiento lingüístico, este se realiza a través de AWS Bedrock, utilizando el modelo Pixtral Large de Mistral, cuyo precio se calcula en función del número de tokens procesados por llamada. En el momento de redacción de esta memoria, el coste estimado para este modelo es de:

- 0,00045 USD por 1.000 tokens de entrada, y
- 0,00158 USD por 1.000 tokens de salida[4].

Cada chunk enviado al modelo contiene entre 2.000 y 3.000 tokens de entrada, y la respuesta generada oscila entre 500 y 1.200 tokens, por lo que el coste promedio por inferencia (es decir, por chunk) se sitúa entre 0,002 y 0,005 USD. Como cada documento se divide en múltiples chunks (dependiendo de su extensión), el coste promedio total por documento procesado ronda los 0,05 a 0,15 USD, lo cual es competitivo frente a otras soluciones LLM del mercado, especialmente teniendo en cuenta la precisión y privacidad ofrecidas.

En lo que respecta a la latencia del sistema, el tiempo de respuesta global depende de la longitud del texto, el número de chunks resultantes y el grado de paralelización habilitado. Gracias a la implementación con ThreadPoolExecutor, el sistema es capaz de procesar múltiples chunks en paralelo, reduciendo drásticamente los tiempos de espera. Por ejemplo, un documento dividido en 80 chunks puede ser procesado en aproximadamente 12 a 18 segundos, frente a más de 60 segundos en una ejecución secuencial. La latencia por chunk individual ronda los 3 a 5 segundos incluyendo tiempo de red, inferencia y parseo de la respuesta. Sin embargo, esta mejora en el tiempo de procesamiento no es gratuita: un mayor número de hilos concurrentes (definido por $MAX_WORKERS$) incrementa el número de llamadas simultáneas a Bedrock, lo que puede generar una subida proporcional de costes si no se limita adecuadamente en función de la carga prevista. Además, Bedrock puede aplicar mecanismos de limitación de tasa (throttling) si se superan los umbrales por segundo configurados por defecto, por lo que es importante dimensionar el sistema de forma equilibrada. Desde una perspectiva de escalabilidad, el sistema demuestra ser apto para producción en escenarios reales. Su arquitectura sin servidor permite absorber picos de carga sin necesidad de redimensionamiento manual, y su estructura modular facilita tanto la facturación granular como la monitorización por cliente o proyecto. En escenarios de uso intensivo, se podrían

aplicar estrategias de optimización adicionales, como la cacheado de inferencias o la agrupación por tipo de documento, para reducir redundancias y mejorar el rendimiento económico.

4.4.8 Selección del modelo y rendimiento

Uno de los elementos clave en el diseño del sistema de detección y anonimización de datos sensibles ha sido la elección del modelo de lenguaje que se encargaría de interpretar el contenido del documento y devolver la información estructurada en formato JSON. Dado que la plataforma de despliegue elegida ha sido AWS Bedrock, se ha contado con una gama de modelos fundacionales de distintos proveedores, entre los que destacan:

- Claude de Anthropic, modelos diseñados con un enfoque en seguridad y robustez conversacional.
- Llama de Meta, modelo diseñado para sistemas de agentes, chat optimizado y generación de código.
- Titan Text de AWS, un modelo optimizado para tareas generales de procesamiento de lenguaje natural.
- Mistral (Pixtral Large), desarrollado por la startup europea Mistral AI, con especial foco en eficiencia y rendimiento técnico.

La selección del modelo final se realizó tras una serie de pruebas comparativas que evaluaron tres factores fundamentales: precisión en la detección de datos sensibles, consistencia estructural de la salida JSON, y coste por inferencia. Se diseñaron pruebas controladas en las que se enviaban los mismos párrafos a distintos modelos, utilizando plantillas de prompt idénticas, y se comparaban los resultados en términos de exhaustividad, clasificación correcta de entidades, y robustez frente a ambigüedades. El único modelo que consiguió una una consistencia del 100% en el formato de la respuesta fue el modelo Pixtral Large de Mistral, garantizando así un correcto funcionamiento de la infraestructura en cada petición. Además, cuenta con la ventaja que es más eficiente en términos económicos que sus competidores directos (Claude y Llama). En términos de escalabilidad, debido a su naturaleza europea, los modelos desarrollados por Mistral son especialmente eficientes en tareas multilingüisticas proponiendo así la posibilidad de aumentar la detección de datos sensibles de documentos redactados en castellano a documentos redactados en inglés, francés, alemán, portugués e italiano. Por último, cabe destacar el excelente rendimiento del modelo. Las pruebas realizadas con diferentes contratos reales, de diferentes contextos del mundo jurídico, fueron solventados sin problema

alguno. Al principio se detectaba cerca de un 98% de los datos sensibles, pero reajustando un poco la lógica de procesado (incluyendo las dos fases de procesado y optimizando la plantilla del prompt) se antepuso la posibilidad de existencia de falsos positivos frente a la de falsos negativos, incluyendo una herramienta de verificación dentro del Addin para validar los datos encontrados por el modelo. Esta funcionalidad será explicada más en adelante en la funcionalidad del AddIn.

4.5 Integración final en Word y descripción del funcionamiento del AddIn

Una vez diseñada la arquitectura ganadora se integra dentro del AddIn de Word para comprobar en un contexto real las capacidades de detección del sistema. La estructura de archivos de la aplicación web definitiva del AddIn se divide en las siguientes partes:

- *taskpane.html (.js)*: Contiene el menú principal con el logo de ClaudIA y los botones para moverte a las dos funcionalidades: **Sensible Information** y **Chat with ClaudIA**.

- *detector.html (.js)*: Expone la interfaz y el funcionamiento del proceso de anonimización y gestiona las peticiones a la API desplegada en AWS.

- *claudIAChat.html (.js)*: Implementa la funcionalidad de ChatBot especializado en tareas de redacción de documentos.

A continuación incluyo una breve descripción en alto nivel basada en la metodología OPM (*Object Process Methodology*) impulsada por el MIT (Massachusetts Institute of Technology) para la estandarización de la representación de modelos industriales, flujos de trabajo, sistemas operativos, etc. La descripción real se puede encontrar en el Anexo que contiene los esquemas del anonimizador y del CLM.

Este complemento 4.2 es utilizado por el usuario para analizar documentos contractuales que pueden contener información sensible, junto con fundamentos legales y normativas.

El proceso de limpieza del contrato ("Cleaning Contract") se inicia cuando el usuario interactúa con el complemento. Este complemento extrae los párrafos del contrato, identifica información sensible y la sustituye por datos anonimizados. Para realizar esta tarea, el sistema se apoya en una arquitectura en la nube provista por AWS descrita anteriormente.

Una vez extraídos los párrafos relevantes, el complemento envía la información al backend a través de una conexión Wi-Fi y una pasarela API (API Gateway).

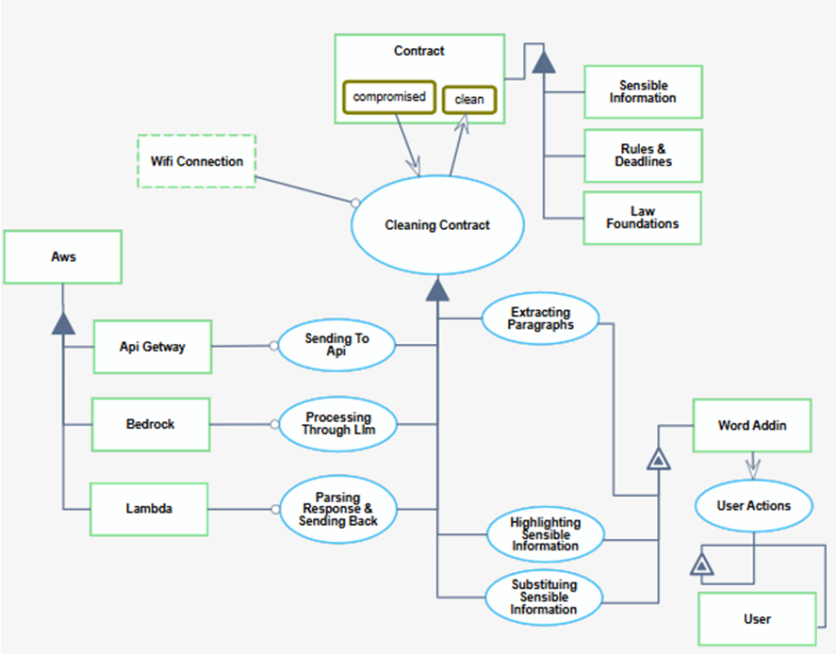


Figure 4.2: Sistema de anonimización representado mediante metodología OPM

Desde ahí, el texto es procesado por un modelo de lenguaje grande (LLM) mediante el servicio Bedrock, el cual analiza el contenido y genera una respuesta con anotaciones o transformaciones necesarias. Esta respuesta es posteriormente gestionada por una función Lambda, que la analiza y la devuelve al complemento.

El resultado final es un contrato limpio, donde la información sensible ha sido identificada y sustituida automáticamente según reglas y criterios legales preestablecidos. Todo este flujo ocurre de manera integrada entre el complemento de Word y los servicios en la nube, facilitando la protección de datos en documentos legales.

4.5.1 Lógica interna del proceso de extracción de párrafos y anonimización del documento

A continuación se presenta una descripción estructurada, formal y clara del funcionamiento del archivo detector.js, el cual forma parte de un complemento de Word basado en Office.js. Este script tiene como objetivo detectar, resaltar, anonimizar y eventualmente revertir datos sensibles en documentos Word mediante el uso de una API basada en modelos de lenguaje.

1. Inicialización del complemento: La función *Office.onReady()* se encarga de verificar que el complemento se está ejecutando dentro de Microsoft Word y

posteriormente asocia funciones a botones del DOM que permiten iniciar la búsqueda, sustitución y reversión de datos sensibles:

```
Office.onReady((info) => {
  if (info.host === Office.HostType.Word) {
    document.getElementById("sensibleSearchButton").onclick =
    handleSensibleSearch;
    document.getElementById("replaceKeywordsButton").onclick
   = replaceSensitiveWords;
    document.getElementById("sensibleSearchSectionButton").
   onclick = handleSensibleSearchSelected;
    document.getElementById("revertAnonymizationButton").
   onclick = revertAnonymization;
    toggleReverseButton(false); // Inicialmente oculto
  }
});
```

Listing 4.5: Inicialización en Word

2. Extracción del contenido del documento: El complemento permite trabajar tanto con el texto completo como con el texto seleccionado por el usuario. Para ello se definen funciones como:

   - *getAllParagraphs()*: Extrae y devuelve todos los párrafos del documento como un array de textos individuales:

```
        const paragraphs = context.document.body.
   paragraphs;
        paragraphs.load("items");

```

Listing 4.6: Extracción de todos los párrafos

   - *getSelectedText()*: Extrae únicamente el texto actualmente seleccionado por el usuario:

```
        const selection = context.document.getSelection()
   ;        selection.load("text");

```

Listing 4.7: Extracción del fragmento seleccionado

3. Comunicación con la API de detección: El texto obtenido se envía al endpoint desplegado en AWS mediante Gateway a través de la función *sendTextToEndpoint()*.

4. Procesamiento de la respuesta: La función *processAPIResponse()* analiza la estructura de la respuesta, separando las palabras clave detectadas por

categorías y almacenándolas en dos variables globales: *keyWords* y *sensitiveDataDict*.

```
1     for (const category in parsedResponse) {
2     if (Array.isArray(parsedResponse[category]) &&
      parsedResponse[category].length > 0) {
3           sensitiveDataDict[category] = parsedResponse[category
      ];
4           keyWords.push(...parsedResponse[category]);
5           }
6     }
7
```

Listing 4.8: Procesamiento de la respuesta

5. Resaltado de palabras sensibles: Una vez detectadas las palabras clave, se ejecuta *highlightSensitiveWords()* para aplicar un resaltado amarillo a cada una dentro del documento Word:

```
1     const searchResults = context.document.body.search(word,
      {
2        matchWholeWord: false,
3        matchCase: false,
4        ignorePunct: true,
5        ignoreSpace: true,
6     });
7     result.font.highlightColor = "yellow";
8
```

Listing 4.9: Resaltado de PII

6. Interfaz interactiva con el usuario de validación: Las palabras sensibles encontradas se presentan visualmente en la interfaz del complemento, con la opción de navegar entre ocurrencias mediante badges que permiten saltar entre cada aparición de un término en el documento (*navigateToOccurrence()*), además de poder eliminar individualmente ciertas palabras del listado y del resaltado.

7. Anonimización del contenido: El módulo ofrece tres métodos para anonimizar las palabras sensibles detectadas:

   - Sustitución por identificadores únicos ([nombre_1], [nombre_2], [dni_1], [dni_2] etc.).
   - Sustitución genérica por asteriscos ([*]).
   - Sustitución por la etiqueta [**CONFIDENCIAL**].

Esto se gestiona mediante la función *replaceSensitiveWords()*, que utiliza el siguiente mapeo:

```
    const anonymizedId = `${normalizedCategory}_${index +
   1}`;
    anonymizationMap[word] = anonymizedId;
    reverseAnonymizationMap[anonymizedId] = word;

```

Listing 4.10: Sustitución de PII

8. Reversión de la anonimización: La función *revertAnonymization()* permite restaurar el contenido original sustituyendo los identificadores por las palabras reales:

```
    const searchPattern = `[${anonymizedId}]`;
    result.insertText(originalWord, Word.InsertLocation.
   replace);

```

Listing 4.11: Reversión de PII

   El proceso es seguro siempre que el reverseAnonymizationMap esté disponible y actualizado.

Este archivo detector.js implementa un sistema completo de detección, visualización, anonimización y restauración de datos sensibles dentro de documentos Word, utilizando la API de Office.js y un backend con inteligencia artificial.

4.5.2 Lógica interna del módulo de interacción con el modelo de lenguaje

A continuación se describe la estructura y funcionamiento del archivo *claudIAChat.js*, el cual integra una interfaz conversacional con un modelo de lenguaje (LLM) en un complemento para Microsoft Word. Este script gestiona la selección de texto, construcción del prompt, envío a un endpoint en la nube y visualización de la respuesta, utilizando la API de Office.js.

1. **Inicialización del complemento:** Se configura el entorno cuando Word está listo, añadiendo manejadores de eventos a los elementos del DOM que permiten seleccionar prompts, cambiar idioma y enviar texto al modelo:

```
Office.onReady((info) => {
  if (info.host === Office.HostType.Word) {
    document.getElementById("prompt-dropdown").
   addEventListener("change", handlePromptDropdownChange);
```

```
4     document.getElementById("language-dropdown").
      addEventListener("change", handleLanguageChange);
5     document.getElementById("submitToLLMButton").
      addEventListener("click", handleSubmitToLLM);
6     ...
7   }
8 });
9
```

Listing 4.12: Inicialización de la extensión

2. **Selección de texto:** Se actualiza periódicamente la variable `selectedText` mediante `getSelectedText()`, función que accede a la selección activa en el documento y la muestra junto con el recuento de palabras:

```
1 const selection = context.document.getSelection();
2 selection.load("text");
3 await context.sync();
4 const text = selection.text;
5
```

Listing 4.13: Extracción de texto seleccionado

3. **Construcción del prompt:** El usuario puede elegir entre plantillas predefinidas (resumen, mejora, traducción, etc.) o ingresar un prompt personalizado. Estos prompts se almacenan en el objeto `promptTemplates` y se concatenan con el texto seleccionado:

```
1 const promptTemplates = {
2   "summary": {
3     "es": "Responde  nicamente  con un resumen conciso del
      siguiente texto...\n\n"
4   },
5   ...
6 };
7
```

Listing 4.14: Estructura de plantillas por idioma

4. **Comunicación con el modelo:** La función `sendToLLM()` envía el prompt completo a un endpoint desplegado en AWS API Gateway. Este endpoint invoca un modelo LLM comercial o privado que devuelve una respuesta procesada:

```
1 const endpointUrl = "https://.../ClaudIaDevEurope";
2 const requestBody = {
3   body: {
4     texto: prompt.trim(),
```

```
    function: "claudia",
  },
};
const response = await fetch(endpointUrl, requestOptions);

```

Listing 4.15: Llamada al endpoint LLM

5. **Visualización e inserción de respuesta:** La respuesta generada por el modelo se inserta en el documento con el mismo formato que el texto original. Se simula un efecto tipo "streaming" al insertar palabra por palabra con retardo:

```
const insertedRange = currentRange.insertText(chunk, Word.
    InsertLocation.end);
insertedRange.font.name = originalFormat.fontName;
insertedRange.font.size = originalFormat.fontSize;
...
await new Promise(resolve => setTimeout(resolve, Math.random
    () * 20 + 10));

```

Listing 4.16: Inserción de respuesta en Word

6. **Interacción con el usuario:** Las funciones `addUserMessage()` y `addAssistantMessage()` gestionan la visualización tipo chat de los mensajes enviados y recibidos. Además, es posible copiar la respuesta o eliminarla del documento.

Este módulo complementa al sistema de anonimización al facilitar el acceso conversacional a funciones de procesamiento de texto dentro del propio entorno de Word, delegando la lógica semántica y lingüística a un modelo LLM externo.

Sistema de Contract Life Management (CLM)

A medida que los contratos se vuelven más complejos, su gestión exige una mayor inversión de recursos humanos y técnicos. Esto implica contar con profesionales capaces de comprender en profundidad la estructura contractual, interpretar cláusulas legales específicas y realizar un seguimiento preciso de hitos y plazos que, en muchos casos, se extienden durante varios años. En este contexto, los sistemas de Contract Lifecycle Management (CLM) adquieren una relevancia fundamental, ya que permiten automatizar tareas críticas como el seguimiento de compromisos contractuales y la generación de calendarios detallados con todos los eventos y obligaciones que deben ser atendidos. Gracias a ello, se reduce significativamente el riesgo de incumplimientos y se mejora la eficiencia operativa en la administración de contratos complejos. El sistema desarrollado combina técnicas de RAG con sistemas de agentes para verificar la capacidad de generar los calendarios contractuales de forma autónoma. En este TFG, los contratos sometidos a análisis son contratos EPC (*Engineering, Procurement and Construction*, por sus siglas en inglés).

5.1 Conceptos Básicos

5.1.1 Retrieval Augmented Generation (RAG)

La arquitectura Retrieval-Augmented Generation (RAG) representa un enfoque híbrido que combina técnicas clásicas de recuperación de información (Information Retrieval, IR) con modelos generativos de lenguaje natural (Large Language Models, LLMs). Su finalidad es potenciar la capacidad de los modelos generativos al proporcionarles acceso dinámico a conocimiento externo, permitiendo así respuestas más precisas, actualizadas y fundamentadas en información factual. Este

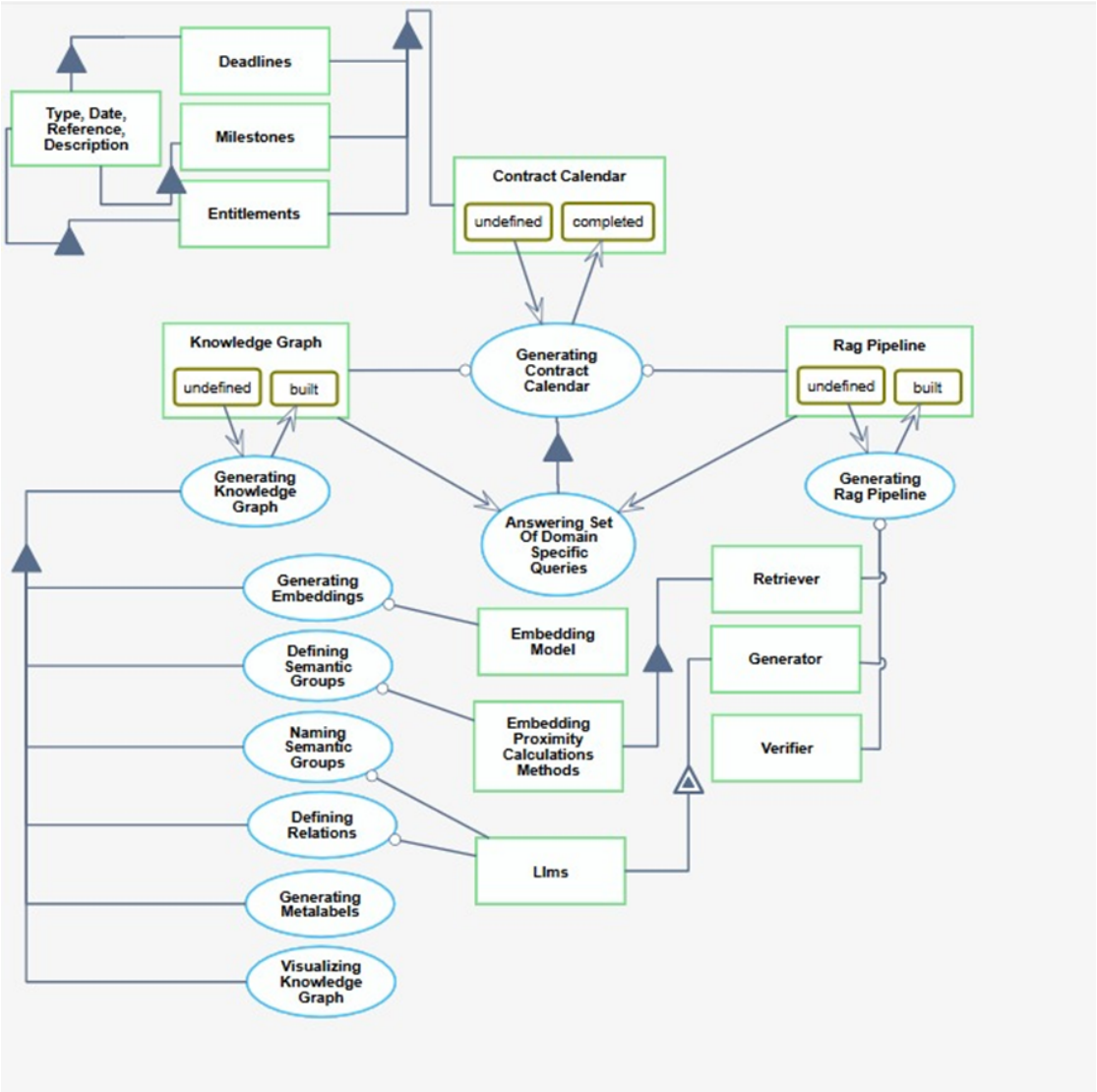


Figure 5.1: Arquitectura CLM representada mediante metodología OPM

paradigma fue introducido formalmente por Lewis et al.[32] como una solución a una limitación inherente a los LLMs: la dependencia exclusiva de los parámetros del modelo para almacenar conocimiento. En dominios intensivos en información, como el jurídico, médico o técnico, los modelos preentrenados pueden carecer de la precisión o actualidad necesarias para generar respuestas fiables. RAG resuelve esta limitación integrando un mecanismo de búsqueda de documentos relevantes a partir de una base de conocimiento externa, comúnmente indexada en una estructura de vectores semánticos (vector store), los cuales son generados a partir de embeddings. El flujo básico de una arquitectura RAG se divide en dos componentes principales:

- **Retriever** (Recuperador de contexto): Dada una consulta o prompt, esta etapa emplea técnicas de recuperación semántica (como búsqueda por similitud de vectores usando FAISS, Weaviate o similares) para identificar los fragmentos de texto más relevantes desde una base documental. Esta base puede estar compuesta por documentos, cláusulas contractuales, artículos científicos, entre otros.
- **Generator** (Generador de texto): Una vez recuperados los documentos pertinentes, estos se concatenan con el input original y se suministran como contexto adicional al modelo generativo, que produce una respuesta enriquecida y contextualizada. Este modelo puede estar basado en arquitecturas como BART, T5 o LLaMA, y opera de forma end-to-end, generando el texto final con una mayor fundamentación en la información extraída.

A diferencia de los sistemas tradicionales de pregunta-respuesta basados únicamente en recuperación (por ejemplo, sistemas tipo BM25 + extractive QA), RAG permite respuestas generativas, libres y adaptadas al estilo del modelo, pero ancladas en evidencia textual relevante. Este enfoque lo convierte en una herramienta clave en sistemas open-domain QA, asistentes conversacionales, gestores de conocimiento corporativo y, de forma creciente, en soluciones LegalTech para análisis y comprensión contractual.

5.1.2 Knowledge Graph (KG)

Un Knowledge Graph (KG), o grafo de conocimiento, es una representación semántica y estructurada del conocimiento que modela entidades (nodos) y las relaciones que existen entre ellas (aristas), formando una red orientada que refleja cómo se conectan conceptualmente los elementos de un dominio específico. Esta estructura permite no solo almacenar información, sino también razonar sobre ella, detectar dependencias implícitas y realizar inferencias que no son directamente observables en los datos de origen.

En términos generales, los grafos de conocimiento integran conceptos de la Web Semántica, bases de datos orientadas a grafos y técnicas de representación del conocimiento provenientes de la inteligencia artificial simbólica. Su fortaleza reside en su capacidad para representar información compleja de manera interpretable, interoperable y extensible, lo cual los convierte en herramientas clave en aplicaciones como motores de búsqueda semántica, sistemas de recomendación, análisis de relaciones en datos empresariales, y recientemente, en sistemas de gestión documental y contractual.

En el ámbito del Contract Lifecycle Management (CLM), los grafos de conocimiento ofrecen una solución especialmente adecuada para abordar la complejidad semántica y estructural de los contratos jurídicos. A través de esta representación, es posible descomponer, mapear y organizar las cláusulas contractuales y sus relaciones de una manera que favorece tanto la automatización como el análisis contextual.

Concretamente, un grafo de conocimiento aplicado a contratos puede representar:

- Cláusulas contractuales como nodos, encapsulando información como el tipo de cláusula (obligación, derecho, condición, penalización, etc.), su contenido textual, y metadatos relevantes (por ejemplo, fecha de activación, parte responsable, etc.).
- Relaciones entre cláusulas como aristas dirigidas, que capturan dependencias lógicas (por ejemplo, "X entra en vigor si Y se cumple"), relaciones de jerarquía (por ejemplo, cláusulas subordinadas) o vínculos cronológicos (por ejemplo, "debe ejecutarse antes de").
- Agrupaciones semánticas como entidades compuestas o subgrafos, que permiten estructurar el contrato en secciones lógicas (por ejemplo, "Obligaciones del proveedor", "Condiciones de terminación", "Régimen sancionador"), facilitando el análisis temático y la navegación modular.

Este tipo de modelado no solo mejora la trazabilidad y comprensión del contrato, sino que también habilita operaciones automatizadas como la detección de inconsistencias, la evaluación del cumplimiento de condiciones contractuales o la generación de cronogramas dinámicos basados en relaciones dependientes entre cláusulas.

5.2 Diseño del Knowledge Graph

Todos los códigos relacionados con el KG y con la pasarela de RAG se pueden encontrar en el repositorio de github presente en el Anexo B 10.

5.2.1 Generación de representaciones vectoriales (embeddings)

Uno de los pasos fundamentales para implementar un sistema RAG (Retrieval-Augmented Generation) consiste en la transformación de texto contractual en representaciones vectoriales que capturen su contenido semántico. Para ello, se ha desarrollado un script en Python que automatiza la extracción, segmentación y vectorización de párrafos contractuales relevantes. Las bibliotecas utilizadas son las siguientes:

- *os y json*: utilizadas para gestionar rutas de archivo y para serializar los resultados en formato JSON.
- *tqdm*: empleada para mostrar una barra de progreso durante el proceso de cálculo de embeddings.
- concurrent.futures: permite la ejecución concurrente de múltiples tareas, en este caso, el cálculo paralelo de embeddings para mejorar el rendimiento del sistema.

La extracción y segmentación del corpus empieza por la función *extracting_and_chunking()* que se encarga de leer el contrato dispuesto a analizar, filtrar las líneas vacías o irrelevantes, y agrupar los párrafos de dos en dos para construir los llamados chunks de texto. Este proceso busca capturar mayor contexto semántico por fragmento, mejorando potencialmente la calidad de los embeddings:

```
for i in range(0, len(content_clean), 2):
    buffer.append(" ".join(content_clean[i:i+2]))
```

Listing 5.1: Generación de chunks

Una vez generados los chunks obtenemos los embeddings mediante la función *get_embedding()* que se conecta al endpoint de embeddings de OpenAI mediante un cliente previamente autenticado. Por defecto, se utiliza el modelo text-embedding-3-small, que ofrece una buena relación entre precisión y coste computacional.

```
resp = client.embeddings.create(input=[clean_text], model=model)
```

Listing 5.2: Generación de Embeddings

. Para reducir el tiempo de espera se implementa la función *calculate_chunk_embeddings()* que orquesta el cálculo de embeddings de forma concurrente, utilizando hasta 10 hilos de ejecución paralela mediante *ThreadPoolExecutor*. Esta estrategia reduce significativamente el tiempo total de procesamiento:

```
with concurrent.futures.ThreadPoolExecutor(max_workers=max_workers) as executor:
    results = list(tqdm(executor.map(embed_chunk, enumerate(buffer)), total=len(buffer)))
```

Listing 5.3: Multiples hilos

Cada chunk embebido se guarda en una lista de diccionarios con las claves id, text y embedding. Finalmente, esta estructura se almacena en un archivo JSON (*chunk_embeddings.json*), que servirá como base vectorial para el posterior sistema de recuperación del módulo RAG.

5.2.2 Cálculo de distancias semánticas

La función *calculate_distances()* se encarga de evaluar la similitud entre todos los pares posibles de textos mediante la similitud coseno, una métrica ampliamente empleada en el procesamiento de lenguaje natural:

```
similarity_matrix = cosine_similarity(X)
distance = 1 - sim
```

Listing 5.4: Cálculo de similitud coseno

Se produce como salida un archivo JSON que contiene las distancias semánticas entre párrafos seleccionados. Este análisis permite obtener una primera aproximación a la cohesión temática del corpus contractual. La función principal del módulo, *agrupamiento_semantico()*, emplea el algoritmo HDBSCAN sobre una matriz de distancias coseno precomputada. Este algoritmo no requiere definir a priori el número de *clusters* y es capaz de identificar automáticamente tanto regiones densas (grupos temáticamente homogéneos) como puntos de ruido (párrafos no relacionados con ningún grupo):

```
clusterer = hdbscan.HDBSCAN(
    metric='precomputed',
    min_cluster_size=min_cluster_size,
    min_samples=min_samples,
    cluster_selection_method=cluster_selection_method,
    allow_single_cluster=False
)
```

Listing 5.5: Agrupamiento con HDBSCAN

Los parámetros del algoritmo han sido cuidadosamente ajustados para maximizar la granularidad semántica de los grupos detectados. En particular, la reducción de *min_cluster_size* y *min_samples* permite detectar agrupaciones de cláusulas incluso cuando estas aparecen de forma escasa pero significativa. Como resultado del proceso, se genera un archivo JSON con grupos de párrafos indexados. Los grupos identificados incluyen tanto conglomerados temáticos (*group_n*) como

instancias individuales que no han sido agrupadas (*group_loneliner_i*). Cada grupo contiene:

- Los índices originales de los párrafos dentro del corpus.
- El texto correspondiente a cada uno.
- Un nombre identificador del grupo.

Este resultado es esencial para el siguiente módulo, en el que se construirá un grafo de conocimiento. Los nodos del grafo corresponderán a los grupos semánticos, mientras que las relaciones se derivarán a partir de patrones comunes detectados entre los mismos.

5.2.3 Etiquetado semántico automático de grupos de cláusulas

Tras haber agrupado semánticamente los párrafos contractuales mediante *clustering* no supervisado, este módulo tiene como objetivo asignar una etiqueta resumida a cada grupo, capturando de forma concisa el contenido jurídico común que comparten sus miembros. Esta etiqueta servirá como identificador semántico de los nodos en el grafo de conocimiento.

Las bibliotecas utilizadas son las siguientes:

- *json, os, time y concurrent.futures*: utilizadas para gestión de archivos, cronometraje de ejecución y paralelización del procesamiento por grupos.
- *tqdm*: empleada para mostrar progreso de ejecución en tiempo real.
- *openai*: permite invocar modelos de lenguaje avanzados a través de la API oficial, para realizar tareas de resumen y clasificación semántica.

La lógica del módulo gira en torno al uso de modelos de lenguaje de propósito general (LLMs), en este caso desplegados mediante la API de OpenAI, para generar etiquetas representativas de cada grupo. El enfoque empleado es *RAG* (Retrieval-Augmented Generation) con generación controlada por *prompt engineering*. La función *build_prompt()* construye dinámicamente un prompt en inglés especializado, que presenta hasta cinco párrafos de un mismo grupo al modelo. Se le indica que actúe como asistente legal especializado en contratos EPC y que devuelva únicamente una etiqueta resumen en lenguaje jurídico:

```
1 "Use 2 to 6 words.\nUse formal and specific legal language (e.g.,
    'Force Majeure Clauses', 'Warranty Obligations')"
```

Listing 5.6: Prompt especializado para generación de etiquetas

Este control sobre el formato permite obtener etiquetas uniformes y aptas para ser utilizadas directamente como nombres de nodo. La función *generate_titles()* filtra todos los grupos con más de una cláusula y los somete a procesamiento paralelo utilizando un *ThreadPoolExecutor*. Cada grupo es enviado individualmente a la función *process_single_group()*, que lanza la llamada al modelo e interpreta su respuesta:

```
response = client.responses.create(
    model=model,
    input=[
        {"role": "system", "content": "..."},
        {"role": "user", "content": prompt}
    ],
)
```

Listing 5.7: Llamada a modelo LLM para etiquetado

Este diseño modular permite realizar inferencias a gran escala, reduciendo significativamente el tiempo total de ejecución. Como paso final, el script actualiza el archivo *semantic_groups.json*, sustituyendo el campo *"group_name"* por la nueva etiqueta generada para cada grupo. Se conservan los grupos originales que no cumplan los criterios mínimos, y se informa al usuario del número total de grupos etiquetados:

```
group_data["group_name"] = group_labels[indices_set]
```

Listing 5.8: Asignación de etiquetas a grupos

Esta salida enriquecerá los nodos del grafo semántico no solo con su contenido (cláusulas), sino también con su función legal, facilitando búsquedas, visualización y razonamiento legal automatizado.

5.2.4 Detección de relaciones semánticas entre grupos contractuales

El propósito de este módulo es identificar relaciones funcionales o jurídicas entre los grupos temáticamente homogéneos generados en etapas anteriores del *pipeline*. A diferencia de los módulos previos, aquí se introduce un criterio de selección eficiente que limita las llamadas a modelos de lenguaje (LLMs) únicamente a aquellas combinaciones de grupos que presentan una alta similitud semántica. Esto permite optimizar significativamente los costes de ejecución manteniendo un alto grado de precisión conceptual.

Las bibliotecas utilizadas son las siguientes:

- *itertools y json*: para generar combinaciones de grupos y manejar la persistencia de resultados.

- *numpy y scikit-learn*: utilizadas para el cálculo de centroides y medidas de similitud coseno entre embeddings.

- *concurrent.futures y tqdm*: permiten procesar relaciones en paralelo con seguimiento en tiempo real del progreso.

El núcleo del módulo reside en la función *definir_relaciones()*, que opera bajo un enfoque de filtrado previo por similitud vectorial para evitar invocaciones innecesarias a modelos LLM. Para ello:

- Se calcula un *embedding* promedio (centroide) para cada grupo con tamaño suficiente (*min_chunks*).

- Se evalúa la similitud coseno entre cada par de centroides. Solo se consideran aquellos cuya similitud supera un umbral configurable (*similarity_threshold*), típicamente alto (por defecto, 0.75).

- Solo estas combinaciones seleccionadas son enviadas al modelo LLM para que determine la relación conceptual entre ambos grupos.

Este enfoque prioriza la eficiencia de costes computacionales y económicos, dado que reduce drásticamente el número total de llamadas a modelos comerciales como *gpt-4* u otros equivalentes:

```
if sim >= similarity_threshold:
    relationship_pairs.add((a, b))
```

Listing 5.9: Filtrado por umbral de similitud

Para cada grupo involucrado en una relación, se seleccionan dinámicamente los fragmentos más representativos. Esto se logra mediante la función *get_representative_chunks()*, que calcula el centroide del grupo y selecciona los fragmentos más cercanos a este, priorizando aquellos de mayor longitud por contener más contexto:

```
top_sorted = sorted(..., key=lambda x: len(x[1]), reverse=True)[:
    top_k]
```

Listing 5.10: Selección de fragmentos relevantes

Este mecanismo permite que los *prompts* enviados al modelo contengan solo ejemplos relevantes y contextualmente ricos, reduciendo el ruido y mejorando la precisión de la respuesta. La función *process_relationship()* estructura un *prompt* instructivo en inglés legal para que el modelo genere una etiqueta relacional ontológica, como por ejemplo: *"Contractual Dependency"* o *"Shared Enforcement Clause"*:

```
"Identify and describe the key legal or functional relationship
    between the two groups, using a short, ontological label."
```

Listing 5.11: Instrucción para generación de relaciones

Este diseño busca uniformidad semántica, concisión y reutilización posterior dentro del grafo de conocimiento. Los resultados se almacenan en un diccionario anidado donde cada grupo se relaciona con sus pares semánticamente próximos y la etiqueta correspondiente. Esta estructura se guarda en un archivo JSON (*relations.json*) para su integración directa como aristas etiquetadas en el grafo de conocimiento contractual.

5.2.5 Generación de metadatos unificados por cláusula contractual

Este módulo tiene por finalidad consolidar la información semántica extraída a lo largo del *pipeline* en una estructura homogénea y autocontenida denominada *meta etiqueta*. Cada entrada representa un fragmento contractual individual (*chunk*) e incorpora su representación vectorial, su texto original y los grupos semánticos a los que pertenece. Esta unificación es esencial para alimentar módulos posteriores de análisis relacional, visualización o trazabilidad contractual. La función principal, *construir_meta_etiqueta()*, recibe tres entradas clave:

- *buffer*: lista de textos contractuales previamente segmentados.
- *embeddings*: lista de vectores que representan semánticamente cada fragmento.
- *group_labels*: diccionario que asocia a cada grupo (representado como conjunto de índices) una etiqueta ontológica generada mediante LLMs.

El proceso se inicia con la construcción de un índice inverso que asocia a cada fragmento (*chunk_id*) el conjunto de grupos etiquetados en los que aparece. En caso de que un fragmento no pertenezca a ningún grupo (por ejemplo, si fue identificado como *outlier* por el algoritmo de *clustering*), se le asigna automáticamente al grupo *"loneliners"*:

```
chunk_to_groups[idx].append(label)
...
"groups_related": chunk_to_groups[i] if chunk_to_groups[i] else ["
    loneliners"]
```

Listing 5.12: Asignación de grupos a fragmentos

Cada fragmento es representado como un diccionario estructurado, que incluye:

- Un identificador sintético del contrato (*id_contract*), útil para simulaciones o entornos de múltiples documentos.
- El *embedding* correspondiente.
- Los metadatos: texto original y grupos semánticos relacionados.

Esta estructura permite establecer fácilmente relaciones entre cláusulas, extraer subgrafos temáticos o visualizar clústeres de significado jurídico. Finalmente, se guarda la estructura en disco bajo el nombre *meta_labels.json*, archivo que contiene un diccionario indexado por el identificador de cada fragmento. Esta persistencia garantiza compatibilidad con cualquier sistema posterior que consuma los datos del *pipeline*:

```
with open(save_path, 'w', encoding='utf-8') as f:
    json.dump(meta_labels, f, ensure_ascii=False, indent=2)
```

Listing 5.13: Guardado de meta etiquetas

5.2.6 Generación y visualización del grafo de conocimiento contractual

El módulo *generate_kg.py* constituye la fase final del sistema de análisis semántico, en la que se genera un grafo de conocimiento visual e interactivo a partir de las relaciones previamente identificadas entre grupos de cláusulas contractuales. Esta representación se implementa mediante la librería *pyvis*, que permite construir interfaces dinámicas basadas en la web para el análisis de grafos. El propósito principal es convertir la salida estructurada del sistema (grupos semánticos, metadatos y relaciones) en una representación visual comprensible para usuarios técnicos y no técnicos. Este grafo permite explorar cómo se interconectan los conceptos clave del contrato, identificar núcleos semánticos y rastrear cláusulas individuales (*loneliners*) dentro de un contexto más amplio. A diferencia de muchos sistemas que omiten nodos aislados, este módulo integra de forma explícita los grupos de una sola cláusula (*loneliners*). Se identifican mediante un tratamiento visual diferenciado (forma triangular y color específico), lo cual aporta una visión completa del corpus y evita pérdidas de información relevante:

```
if is_loneliner:
    shape = "triangle"
    color_id = 0
```

Listing 5.14: Tratamiento visual de loneliners

La inclusión de *loneliners* permite detectar cláusulas independientes o excepcionalmente específicas, que podrían ser claves para riesgos jurídicos o excepciones contractuales. La visualización es altamente expresiva:

- El tamaño del nodo depende del número de cláusulas que agrupa.
- El color varía en función del grupo, permitiendo identificar conglomerados temáticos.
- El *tooltip* sobre cada nodo muestra el nombre del grupo y una muestra de los fragmentos que lo componen.
- Las relaciones explícitas (generadas por LLMs) aparecen como aristas con etiquetas jurídicas.
- Las relaciones implícitas entre *loneliners* y grupos se representan con líneas punteadas y color gris, bajo la relación genérica *"part of"*.

Esto facilita una navegación semántica intuitiva del grafo. El módulo también calcula y guarda los *embeddings* semánticos de cada grupo, utilizando la función *get_embedding()* del módulo previo. Esta información puede ser utilizada para análisis adicionales, como *clustering* interactivo, búsquedas por similitud o detección de comunidades:

```
embedding = get_embedding(group_name, client)
```

Listing 5.15: Cálculo de embedding para nodos del grafo

El grafo se guarda en un archivo HTML interactivo (*kgraph.html* por defecto), que puede abrirse en cualquier navegador sin requerir software adicional. Además, el módulo retorna un resumen estructurado con estadísticas clave: número de nodos añadidos, relaciones creadas y cantidad de *loneliners* detectados:

```
return {
    "network": net,
    "group_embeddings": group_embeddings,
    "nodes_added": len(added_nodes),
    "edges_added": edge_count + loneliner_edge_count,
    "loneliner_count": loneliner_count
}
```

Listing 5.16: Resumen estructurado del grafo

Este módulo proporciona una capa crítica de interpretabilidad y navegabilidad al sistema CLM. La generación de un grafo de conocimiento accesible, estéticamente diferenciada y semánticamente coherente facilita la exploración del contenido contractual y refuerza la aplicabilidad práctica del sistema en contextos reales de análisis jurídico.

5.3 Conclusión: Diseño e implementación de un grafo de conocimiento para contratos EPC

El desarrollo del grafo de conocimiento presentado en este proyecto constituye una solución integral y escalable para la representación estructurada del contenido semántico de contratos EPC (*Engineering, Procurement and Construction*). A través de una arquitectura modular y eficiente, se ha logrado transformar un corpus textual desestructurado en un sistema navegable, inteligible y alineado con los principios de la inteligencia artificial legal aplicada.

El *pipeline* diseñado abarca todas las fases necesarias para esta transformación:

- **Vectorización semántica**: mediante la utilización de modelos de *embeddings* avanzados, se ha capturado el contenido latente de cada cláusula contractual. La segmentación por párrafos y el cálculo paralelo de *embeddings* han permitido optimizar tiempos sin comprometer la precisión.

- **Agrupamiento temático**: se implementó un sistema de *clustering* no supervisado (HDBSCAN) que, sin requerir un número de grupos predeterminado, identificó conglomerados de cláusulas con significados afines. Esta fase sentó las bases estructurales del grafo.

- **Etiquetado ontológico**: para dotar de inteligibilidad a cada grupo, se recurrió a modelos de lenguaje generativo que, mediante *prompts* especializados, generaron títulos jurídicos concisos y representativos. Esta fase aportó una capa de interpretabilidad imprescindible para el uso práctico del grafo.

- **Detección de relaciones intergrupales**: se introdujo un enfoque de generación de relaciones basado en umbrales de similitud semántica, lo que permitió reducir el número de inferencias costosas y centrar los esfuerzos computacionales solo en aquellos pares con alta probabilidad de relación jurídica. Esta optimización resultó clave para la escalabilidad del sistema.

- **Consolidación de metadatos**: a través de un modelo de etiquetas unificadas (*meta_labels*), se logró centralizar toda la información relevante —texto, *embedding*, grupo y relaciones— por fragmento, facilitando el acceso, análisis y reutilización de datos.

- **Visualización interactiva**: finalmente, se generó un grafo navegable en formato web que permite explorar visualmente las relaciones contractuales detectadas. Se prestó especial atención a la representación diferenciada de cláusulas aisladas (*loneliners*) y a la semántica visual de nodos y aristas.

En su conjunto, este sistema constituye un avance significativo en la aplicación de técnicas de procesamiento de lenguaje natural y grafos semánticos al dominio legal. Su modularidad permite futuras extensiones —como la incorporación de múltiples contratos, la detección de obligaciones críticas, o la generación automática de cronogramas contractuales—, mientras que su diseño eficiente garantiza viabilidad incluso en contextos con restricciones de recursos computacionales.

Este grafo de conocimiento no solo representa una herramienta técnica, sino también un puente entre el lenguaje jurídico tradicional y las nuevas posibilidades que ofrece la inteligencia artificial para el análisis automatizado de contratos complejos.

5.4 Diseño del Pipeline y generación de domain-specific RAG

La combinación de grafos de conocimiento con arquitecturas RAG representa un enfoque potente y complementario para el tratamiento de información jurídica compleja. Mientras que el sistema RAG permite recuperar información contextual relevante y generar respuestas a partir de un corpus documental, el grafo de conocimiento aporta una capa adicional de estructuración semántica que mejora la comprensión, navegabilidad y precisión del sistema.

En el presente proyecto, el grafo de conocimiento generado a partir del análisis semántico de contratos EPC se integra como una fuente estructurada de contexto especializado en el domain master del sistema RAG. Esta integración permite al modelo no solo recuperar fragmentos textuales relevantes, sino también acceder a conceptos jurídicos, relaciones y agrupaciones semánticas previamente detectadas, mejorando así la calidad de las respuestas y la capacidad de razonamiento sobre el contrato.

5.4.1 Retriever: recuperación híbrida informada por grafo de conocimiento

La clase *HybridRetriever* constituye el componente central de recuperación semántica del sistema RAG especializado. Su diseño combina la recuperación densa por similitud vectorial con una expansión estructurada basada en relaciones del grafo de conocimiento, lo que permite ofrecer al generador de respuestas no solo fragmentos relevantes, sino también contexto jurídico explícitamente relacionado. El módulo carga desde disco todos los artefactos semánticos generados previamente:

- *Embeddings* de párrafos contractuales (`embeddings.json`),

- Grupos semánticos y sus cláusulas (`semantic_groups.json`),
- Relaciones ontológicas entre grupos (`relations.json`),
- Metadatos por fragmento (`meta_labels.json`).

A partir de estos datos, se calculan los *embeddings* de cada grupo como el promedio de los vectores de sus cláusulas asociadas:

```
group_embs[group] = np.mean(vectors, axis=0).tolist()
```

Listing 5.17: Cálculo del embedding de grupo

Este enfoque permite representar cada grupo como un concepto jurídico latente en el espacio vectorial. La función principal *retrieve_context()* ejecuta una estrategia de recuperación en tres fases:

1. Búsqueda densa (semántica): Se calcula la similitud coseno entre la *query* y el *embedding* de cada grupo temático. Se seleccionan los `top_k` grupos más relevantes cuya similitud supere un umbral configurable (`sim_threshold`). Este paso asegura relevancia temática.

2. Refuerzo con cláusulas individuales (*loneliners*): Para mejorar la cobertura y precisión, se seleccionan explícitamente los fragmentos individuales más cercanos a la *query*, incluso si no pertenecen a ningún grupo denso. Esto evita omisiones en casos donde la información relevante esté contenida en cláusulas aisladas:

```
top_loneliners = sorted(loneliner_scores, key=lambda x: x[1],
    reverse=True)[:force_loneliners]
```

Listing 5.18: Selección de cláusulas individuales relevantes

3. Expansión estructural vía grafo de conocimiento: Los grupos seleccionados se expanden con sus vecinos directos en el grafo de relaciones (`relations.json`). Este paso introduce contexto funcional y jurídico adicional, guiado por las conexiones semánticas inferidas en etapas previas:

```
for g in selected_groups:
    neighbors |= set(self.relations.get(g, {}).keys())
```

Listing 5.19: Expansión mediante el grafo de relaciones

Esta integración permite superar la recuperación puramente estadística, incorporando estructuras conceptuales propias del dominio legal. El método retorna una lista de fragmentos relevantes acompañados de sus metadatos, incluyendo el

texto, su ID y los grupos semánticos a los que pertenece. Esta información está lista para ser utilizada por el componente generador del sistema RAG, que puede apoyarse tanto en contenido explícito como en estructura ontológica para generar respuestas fundamentadas.

5.4.2 Generator: generación de eventos estructurados a partir de contexto legal

El módulo `generator.py` constituye el componente generativo del sistema RAG especializado, cuya función principal es transformar una pregunta legal concreta y su contexto documental en un evento contractual estructurado. La salida se encuentra en formato JSON, lo que permite su procesamiento posterior por sistemas de gestión contractual o visualización.

El funcionamiento de este módulo puede dividirse en tres fases:

A partir de los fragmentos recuperados por el módulo *HybridRetriever*, se genera un bloque de texto plano estructurado, que incluye el ID del fragmento, los grupos semánticos asociados y su contenido textual. Esta información se limita a un tamaño máximo configurable para evitar desbordamientos en el *prompt*:

```
block = f"Chunk {c['chunk_id']} (Groups: {', '.join(c['groups'])})
    :\n{c['text']}\n\n"
```

Listing 5.20: Formateo del contexto

El contexto formateado y la *query* del usuario se integran en una plantilla de *prompt* escrita en lenguaje natural mediante la biblioteca *jinja2*. Este diseño desacopla la lógica del sistema de la redacción del *prompt*, facilitando su adaptación y mejora continua sin modificar el código fuente:

```
final_prompt = Template(prompt_template).render(
    context=context_text,
    query=query,
    notice_date="30 June 2023"
)
```

Listing 5.21: Construcción del prompt con Jinja2

Una vez construido el *prompt*, se lanza una consulta al modelo LLM especificado (por defecto, `o4-mini-2025-04-16`) mediante la API de OpenAI. Se espera como respuesta un objeto JSON que representa el evento contractual detectado, incluyendo información como: tipo de evento, fecha crítica, cláusulas implicadas, consecuencias jurídicas, etc.

Se incluyen mecanismos de robustez para interpretar la salida, incluso si está embebida dentro de comillas o contiene errores de codificación JSON:

```
event = parse_llm_json_response(output_text)
```

Listing 5.22: Parseo robusto de la respuesta LLM

La respuesta del modelo debe adoptar una forma estructurada, lo que permite su uso automático en otras partes del sistema. Un ejemplo típico sería:

```
{
  "event_type": "Delay Liquidated Damages Start",
  "trigger_date": "2023-07-01",
  "clauses": ["Clause 8.2", "Clause 15.3"],
  "legal_basis": "Contractual penalty for late delivery",
  "risk_level": "high"
}
```

Listing 5.23: Evento contractual estructurado

Esta estructuración convierte al sistema en algo más que un asistente textual: lo transforma en un motor semántico capaz de extraer conocimiento contractual accionable.

5.4.3 Verifier: verificación de eventos mediante citas textuales

El módulo `verifier.py` cumple una función esencial en la arquitectura del sistema RAG especializado: verificar y justificar los eventos generados por el modelo LLM, encontrando fragmentos del contrato que respalden explícitamente cada uno de sus componentes. Esta verificación aporta trazabilidad, auditabilidad y confianza al usuario final, especialmente en contextos jurídicos donde cada afirmación debe estar sustentada en evidencia documental. El proceso de verificación se aplica tras la generación del evento estructurado (por ejemplo, un *Deadline*, una *Penalty Clause*, etc.) y se realiza exclusivamente sobre los fragmentos previamente recuperados como contexto.

La función principal `verify_event_sources()` recibe:

- Uno o varios eventos estructurados (en formato `dict` o `list[dict]`),
- Una lista de fragmentos de texto (`context_chunks`) extraídos por el `HybridRetriever`.

A partir de ahí, aplica una función de emparejamiento difuso (`match_chunk()`) que compara los campos clave del evento con cada fragmento textual, buscando coincidencias significativas. Los campos considerados para verificación se encuentran definidos como constantes en `EVENT_FIELDS`, e incluyen:

- `name`
- `description`

- `clause_reference`
- `deadline`
- `relative_to_notice`

Cada uno se evalúa con distintas estrategias:

- Coincidencia directa o por expresión regular en el caso de referencias a cláusulas (`clause_reference`),
- Similitud difusa (*fuzzy matching*) basada en `difflib.SequenceMatcher` para campos como `description` o `deadline`.

Esto permite detectar citas relevantes incluso si existen variaciones gramaticales o terminológicas.

Cada campo que coincide incrementa un puntaje acumulado del fragmento. Las coincidencias exactas (como `clause_reference`) reciben mayor peso, reflejando su valor probatorio superior:

```
if field == "clause_reference":
    score += 2  # mayor peso
...
else:
    if ratio >= thresholds[field]:
        score += 1
```

Listing 5.24: Asignación de puntajes por coincidencia

Como salida, cada evento es devuelto con un nuevo campo: `source_citations`, que contiene los `top_k` fragmentos del contrato que mejor justifican su contenido. Para cada fragmento se incluye:

- Su `chunk_id`,
- Un fragmento textual truncado (`text_snippet`),
- Los campos del evento con los que coincide (`match_fields`),
- Un puntaje de relevancia (`score`).

Esto permite al usuario visualizar no solo el evento inferido, sino también los fundamentos textuales que lo respaldan.

5.4.4 Orquestación de la arquitectura RAG + KG

El archivo `run_query.py` implementa la ejecución secuencial y automatizada del *pipeline* completo de recuperación, generación y verificación de eventos contractuales a partir de preguntas jurídicas específicas. Su propósito es transformar consultas legales en eventos estructurados respaldados por citas textuales, y representar estos eventos en un calendario visual que permite su seguimiento operativo. El script organiza el *pipeline* en las siguientes etapas: Se cargan las variables de entorno necesarias (*dotenv*) y se instancian los componentes principales: un cliente de modelo (OpenAI) y el módulo de recuperación semántica `HybridRetriever`. El sistema selecciona los fragmentos contractuales más relevantes para la *query* proporcionada, utilizando una combinación de similitud vectorial, expansión por grafo de conocimiento y refuerzo con cláusulas solitarias (*loneliners*). Esta etapa garantiza máxima relevancia y cobertura contextual:

```
context_chunks = retriever.retrieve_context(
    query=query,
    top_k=TOP_K_GROUPS,
    sim_threshold=SIM_THRESHOLD,
    force_loneliners=NUM_LONELINERS,
    include_neighbors=INCLUDE_NEIGHBORS
)
```

Listing 5.25: Recuperación semántica contextualizada

El contexto recuperado se introduce en un *prompt* personalizado que invoca al modelo de lenguaje para generar una respuesta formalizada. La generación utiliza plantillas *jinja2* para integrar contexto, consulta y fecha de referencia de forma flexible y reutilizable. Cada evento generado es evaluado y enriquecido con citas textuales provenientes del contexto. Este paso, ejecutado por el módulo `verifier.py`, mejora la trazabilidad y confiabilidad jurídica de los resultados. Todos los eventos verificados se agregan a una lista y se almacenan en el archivo `generated_calendar.json`. Finalmente, se genera un calendario HTML interactivo que visualiza los hitos contractuales extraídos, facilitando su análisis temporal:

```
html_content = generate_contractual_calendar_html(calendar_data)
```

Listing 5.26: Generación del calendario contractual

El script permite procesar un conjunto de preguntas definidas en el archivo `queries.json`, ejecutándolas en serie y acumulando los eventos resultantes. Esto permite evaluar de forma masiva un contrato y generar un calendario contractual inteligente, basado en preguntas relevantes para la gestión del proyecto.

5.4.5 Visualización interactiva del calendario contractual

El módulo `generate_calendar_vis.py` transforma los eventos contractuales estructurados, generados y verificados a partir del sistema RAG, en un *dashboard* HTML interactivo. Esta interfaz permite visualizar y explorar los principales hitos del contrato (plazos, derechos, obligaciones, etc.) en formato calendario, ofreciendo una experiencia intuitiva orientada tanto a usuarios legales como a gestores de proyecto. El objetivo principal es ofrecer una capa de visualización enriquecida que permita:

- Filtrar y clasificar eventos por tipo (plazos, hitos, derechos, etc.),
- Buscar eventos por texto libre (nombre, cláusula, descripción...),
- Exportar los resultados a formatos abiertos (JSON y CSV),
- Imprimir directamente el calendario contractual,
- Consultar citas asociadas (cuando estén disponibles).

Todo esto se realiza mediante una única función principal: `generate_contractual_calendar` que genera el contenido HTML completo y lo guarda como archivo local. El módulo recibe como entrada una lista de eventos en formato JSON, típicamente el resultado del *pipeline* RAG tras ejecutar `run_query.py`. Los eventos son limpiados y validados antes de su inclusión en la visualización:

```
1 if isinstance(item, dict) and not item.get('error') and item.get('
    name'):
```

Listing 5.27: Validación de evento antes de su visualización

Cada evento es representado como una "tarjeta" que muestra:

- Su tipo (ej. `"Deadline"`),
- Nombre del evento,
- Fecha crítica o plazo,
- Referencia contractual,
- Descripción resumida,
- Citas textuales justificativas (si existen).

La interfaz presenta un diseño moderno, responsivo y accesible:

- Utiliza un diseño de tarjetas para destacar cada evento.

- Incluye estadísticas dinámicas (número total de eventos, plazos, hitos, etc.).
- Permite aplicar filtros por tipo y realizar búsquedas en tiempo real.
- Ofrece opciones de exportación y compatibilidad con impresión.

El estilo está contenido en la propia plantilla HTML, lo que permite su despliegue inmediato en cualquier navegador sin necesidad de dependencias externas. El módulo incluye también una función `load_calendar_from_file()` para facilitar la carga de datos desde un archivo JSON, lo que permite separar la generación del contenido de su visualización. `generate_calendar_vis.py` cierra el ciclo de análisis documental proporcionando una interfaz visual efectiva y profesional. Su diseño permite comunicar los resultados generados por el sistema de forma clara, ordenada y práctica, ofreciendo así un producto completo que va desde la consulta legal hasta la planificación contractual visual y trazable. Es una muestra clara de cómo la inteligencia artificial, cuando se combina con buenas prácticas de diseño, puede facilitar la gestión jurídica en proyectos de alta complejidad.

5.5 Conclusión: integración del grafo de conocimiento en un sistema RAG especializado

La integración del grafo de conocimiento en el sistema RAG desarrollado representa un avance significativo en la manera en que se puede procesar, estructurar y explotar el contenido jurídico de contratos EPC. Este enfoque híbrido, que combina recuperación semántica, razonamiento estructurado y generación asistida por modelos de lenguaje, ha permitido diseñar una arquitectura robusta, trazable y adaptada al dominio legal.

A diferencia de un sistema RAG convencional, basado únicamente en *embeddings* y recuperación densa, la introducción del grafo de conocimiento aporta una capa semántica superior: permite organizar los fragmentos textuales en grupos temáticos, detectar relaciones funcionales entre cláusulas y enriquecer la recuperación con contexto ontológico. Esta estructuración no solo mejora la relevancia del contenido recuperado, sino que también introduce capacidad de razonamiento explícito, más cercana al análisis jurídico humano.

Por otra parte, la generación de eventos estructurados a partir del contexto recuperado demuestra cómo es posible traducir lenguaje natural en objetos de conocimiento formales, con campos como tipo de evento, plazo, cláusula de referencia o fundamentos legales. Esta transformación habilita la creación de herramientas prácticas, como calendarios contractuales automatizados, que permiten a los usuarios visualizar, exportar y rastrear hitos clave del contrato sin necesidad de leer el documento completo.

El sistema incorpora también un módulo de verificación semántica que garantiza la trazabilidad de las respuestas generadas, al identificar citas textuales en el contrato que respaldan cada evento detectado. Esta funcionalidad es crítica para entornos profesionales donde la confianza, la auditoría y la explicabilidad no son opcionales, sino requisitos fundamentales.

En conjunto, la arquitectura diseñada constituye una solución modular, eficiente y corregible que puede extenderse fácilmente a otros dominios legales o contractuales. Su diseño demuestra que es posible construir sistemas inteligentes que no solo respondan a preguntas, sino que comprendan, estructuren y visualicen el conocimiento jurídico con un alto grado de autonomía y fiabilidad.

Análisis de resultados

6.1 Evaluación del rendimiento del anonimizado

El objetivo de este módulo es proteger la confidencialidad de los datos personales y sensibles contenidos en contratos, mediante su detección y anonimización automática. La herramienta se diseñó para anonimizar entidades conforme al Reglamento General de Protección de Datos (RGPD), sin comprometer la coherencia gramatical o semántica del documento. Por otra parte, se incluirán demostraciones del funcionamiento en el Anexo B **??**, tanto del sistema de anonimización como del chatbot integrado.

6.1.1 Ejemplo de caso real

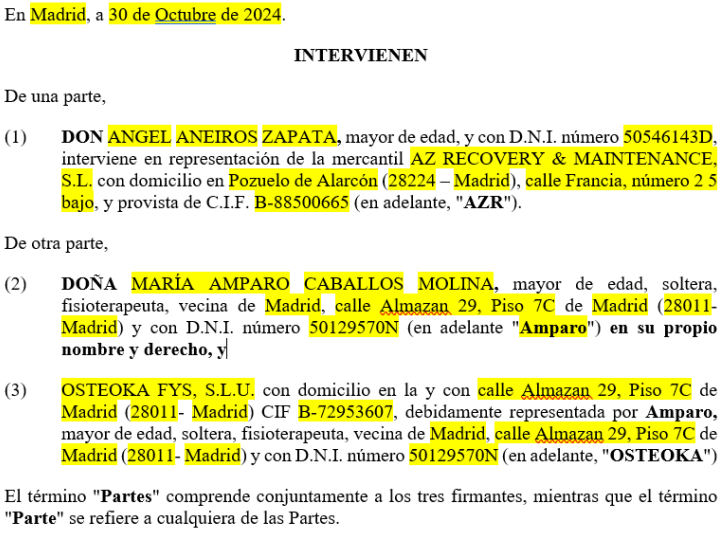
A continuación se muestra un ejemplo representativo de una cláusula legal antes y después de ser procesada por el sistema de anonimización:

Fragmento original (antes de la anonimización)6.1:

> En Madrid, a 30 de Octubre de 2024.
> **INTERVIENEN**
> De una parte,
> (1) DON ANGEL AN*** ZA***, mayor de edad, y con D.N.I. número 50**** (por razones de privacidad), interviene en representación de la mercantil AZ RECOVERY & MAINTENANCE, S.L. con domicilio en Pozuelo de Alarcón, calle Francia, número 2 5 bajo, y provista de C.I.F. B-88500665 (en adelante, "AZR").

Fragmento anonimizado6.2:

> En [ciudades_1], a [fechas_1].
> **INTERVIENEN**



En Madrid, a 30 de Octubre de 2024.

INTERVIENEN

De una parte,

(1) **DON** ANGEL ANEIROS ZAPATA**,** mayor de edad, y con D.N.I. número 50546143D, interviene en representación de la mercantil AZ RECOVERY & MAINTENANCE, S.L. con domicilio en Pozuelo de Alarcón (28224 – Madrid), calle Francia, número 2 5 bajo, y provista de C.I.F. B-88500665 (en adelante, "**AZR**").

De otra parte,

(2) **DOÑA** MARÍA AMPARO CABALLOS MOLINA**,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante "**Amparo**") **en su propio nombre y derecho, y**

(3) OSTEOKA FYS, S.L.U. con domicilio en la y con calle Almazan 29, Piso 7C de Madrid (28011- Madrid) CIF B-72953607, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante, "**OSTEOKA**")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 6.1: PII Marcado

De una parte,
(1) DON [nombres_1] [apellidos_1] [apellidos_2], mayor de edad, y con D.N.I. número [otros_1], interviene en representación de la mercantil [nombres_empresas_1] con domicilio en [ciudades_2], [direcciones_1], y provista de C.I.F. [otros_2] (en adelante, "AZR").

Por último vemos como tras haber anonimizado revertimos la anonimización y lo llevamos al estado inicial 6.3.

6.1.2 Resultados observados

1. Categorías de datos detectadas

El sistema logró identificar y anonimizar múltiples tipos de entidades sensibles, entre ellas:

2. Consistencia contextual

La anonimización mantiene la estructura gramatical del documento intacta. Las entidades sustituidas son reemplazadas por etiquetas que permiten su trazabilidad en caso de reidentificación controlada. Además:

- Se respetaron los pronombres y conectores (por ejemplo: “interviene en representación de...”), evitando errores de sintaxis.

En ***[ciudades_1]***, a ***[fechas_1]***.

INTERVIENEN

De una parte,

(1) **DON** ***[nombres_1] [apellidos_1] [apellidos_2]*****,** mayor de edad, y con D.N.I. número ***[otros_1]***, interviene en representación de la mercantil ***[nombres_empresas_1]*** con domicilio en ***[ciudades_2]*** (***[codigos_postales_1]*** – ***[ciudades_1]***), ***[direcciones_1]***, y provista de C.I.F. ***[otros_2]*** (en adelante, "**AZR**").

De otra parte,

(2) **DOÑA** ***[nombres_2] [apellidos_3] [apellidos_4]*****,** mayor de edad, soltera, fisioterapeuta, vecina de ***[ciudades_1]***, ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) y con D.N.I. número ***[otros_3]*** (en adelante "***[nombres_3]*** ") **en su propio nombre y derecho, y**

(3) ***[nombres_empresas_2]*** con domicilio en la y con ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) CIF ***[otros_4]***, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de ***[ciudades_1]***, ***[direcciones_2]*** de ***[ciudades_1]*** (***[codigos_postales_2]***- ***[ciudades_1]***) y con D.N.I. número ***[otros_3]*** (en adelante, "***[nombres_empresa_3]***")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 6.2: PII Extraído

En Madrid, a 30 de Octubre de 2024.

INTERVIENEN

De una parte,

(1) **DON** Angel Aneiros Zapata**,** mayor de edad, y con D.N.I. número 50546143D, interviene en representación de la mercantil AZ RECOVERY & MAINTENANCE, S.L. con domicilio en Pozuelo de Alarcón (28224 – Madrid), calle Francia, número 2 5 bajo, y provista de C.I.F. B-88500665 (en adelante, "**AZR**").

De otra parte,

(2) **DOÑA** María Amparo Caballos Molina**,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante "**Amparo**") **en su propio nombre y derecho, y**

(3) OSTEOKA FYS, S.L.U. con domicilio en la y con calle Almazan 29, Piso 7C de Madrid (28011- Madrid) CIF B-72953607, debidamente representada por **Amparo,** mayor de edad, soltera, fisioterapeuta, vecina de Madrid, calle Almazan 29, Piso 7C de Madrid (28011- Madrid) y con D.N.I. número 50129570N (en adelante, "**OSTEOKA**")

El término "**Partes**" comprende conjuntamente a los tres firmantes, mientras que el término "**Parte**" se refiere a cualquiera de las Partes.

Figure 6.3: PII Restaurado

Tipo de dato	Ejemplo original	Token anonimizado
Ciudad	Madrid	[ciudades_1]
Fecha	30 de Octubre de 2024	[fechas_1]
Nombre y apellidos	ANGEL AN*** ZAP***	[nombres_1] [apellidos_1] [apellidos_2]
Documento de identidad	5******	[dni_1]
Empresa	AZ RECOVERY & MAINTENANCE, S.L.	[nombres_empresas_1]
Dirección postal	calle Francia, número 2 5 bajo	[direcciones_1]
Código postal	28224	[codigos_postales_1]
C.I.F.	B-88500665	[otros_2]

Table 6.1: Entidades sensibles detectadas y su anonimización

- Las etiquetas están sistemáticamente formateadas con prefijos que permiten su clasificación (`[nombres_]`, `[ciudades_]`, `[otros_]`, etc.).

3. Detección redundante y cruzada

El sistema reconoció instancias repetidas del mismo individuo o entidad (por ejemplo, *"Amparo"*) en diferentes ubicaciones del texto, asegurando que todas las apariciones fueran tratadas de forma coherente.

4. Conservación de alias legales

A pesar de la anonimización de los nombres legales, los alias contractuales definidos entre comillas (por ejemplo, *"AZR"* y *"OSTEOKA"*) no fueron identificados por el modelo. Esto proporciona feedback útil para entender que clase de datos no son reconocibles todavía y cómo mejorar el sistema.

6.1.3 Conclusión

El sistema de anonimización ha demostrado una alta precisión en la detección y sustitución de información sensible, manteniendo la legibilidad y validez del texto contractual. Esto lo convierte en una herramienta adecuada para:

- La distribución segura de contratos en entornos colaborativos.
- El entrenamiento de modelos LLM sobre datos reales sin violar la privacidad.
- La indexación legal anonimizada para sistemas de recuperación o análisis jurídico.

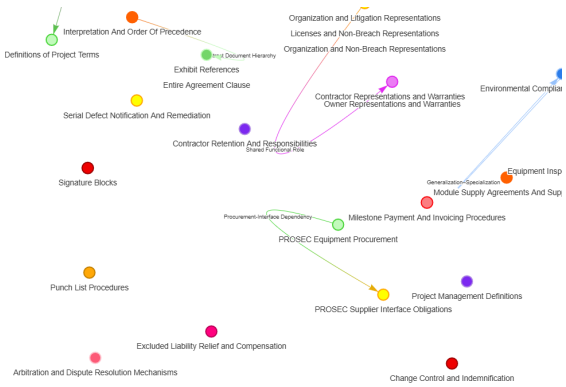


Figure 6.4: Fragmento del KG compuesto por grupos no interconectados

6.2 Evaluación de resultados: grafo de conocimiento aplicado a contratos EPC

El objetivo de esta etapa fue representar de forma estructurada y navegable las relaciones lógicas, temporales y funcionales entre cláusulas de un contrato EPC complejo, con el fin de facilitar el análisis legal asistido por inteligencia artificial. Para ello, se construyó un grafo de conocimiento a partir de los resultados obtenidos en una arquitectura *RAG* (Retrieval-Augmented Generation), complementado con una capa de agrupación semántica.

6.2.1 Agrupación semántica

Para facilitar la abstracción temática del documento, se agruparon los fragmentos textuales extraídos en *clústeres* semánticos. Cada grupo representa una categoría funcional dentro del contrato (por ejemplo: *"Warranties"*, *"Liquidated Damages"*, *"Change Orders"*, *"Subcontractors"*), lo cual permite asociar visualmente las cláusulas a su rol dentro del ciclo de vida contractual.

Se detectaron más de 90 grupos semánticos, cada uno formado por uno o varios párrafos legalmente vinculantes.

Esta agrupación sirvió de base para la codificación visual del grafo, donde cada nodo hereda la categoría semántica como su grupo (`group` en `vis.js`).

6.2.2 Extracción de relaciones

Se utilizaron métodos basados en LLM para detectar relaciones entre cláusulas, clasificadas según su tipo semántico: *dependencia*, *consecuencia*, *generalización-especialización*, *obligación-remedio*, *precondición*, entre otras.

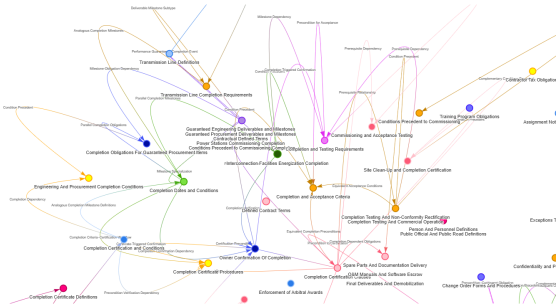


Figure 6.5: Fragmento del KG compuesto por grupos relacionados

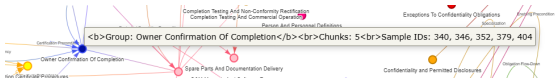


Figure 6.6: Nodo (grupo semántico) con meta información sobre chunks asociados

Se generaron más de 100 relaciones distintas entre nodos, almacenadas en el archivo `relations.json`.

La codificación de las relaciones incluye tanto vínculos bilaterales como jerárquicos, lo que permite detectar estructuras contractuales como escalamientos de remedios, obligaciones secuenciales o dependencias cruzadas entre secciones aparentemente disjuntas.

6.2.3 Visualización interactiva

El grafo fue renderizado en una interfaz web (`kgraph.html`) utilizando la librería `vis-network`. Se incorporaron funcionalidades de navegación y filtrado que permiten:

- Resaltar vecinos directos e indirectos de un nodo (hasta 2 grados).
- Filtrar por propiedades específicas (por ejemplo, cláusulas con remedios económicos).
- Explorar nodos por grupo semántico, facilitando el análisis de áreas contractuales.

6.2.4 Observaciones y valoración

El grafo revela de manera efectiva las interdependencias entre cláusulas críticas como garantías, penalizaciones por retrasos y condiciones de terminación.

Las relaciones de tipo *"Shared Trigger Event"* y *"Escalation of Remedies"* muestran una lógica contractual secuencial útil para automatizar análisis de riesgos o simulaciones legales.

La representación gráfica mejora la trazabilidad y reduce la carga cognitiva en la revisión de contratos extensos, especialmente en entornos colaborativos legales y de *compliance*.

6.2.5 Conclusiones

La generación del grafo de conocimiento ha cumplido con el objetivo de representar estructuralmente el contenido legal de un contrato EPC de gran complejidad. La combinación de análisis semántico y visualización relacional sienta las bases para aplicaciones futuras como:

- Generación de calendarios automáticos de hitos contractuales.
- Evaluación del cumplimiento de obligaciones.
- Análisis predictivo de eventos de incumplimiento.

6.3 Evaluación de resultados: generación del calendario contractual

Objetivo

Esta fase del proyecto tuvo como finalidad construir un calendario estructurado de eventos contractuales, permitiendo la visualización cronológica y semántica de plazos, hitos, derechos y obligaciones derivadas del contrato EPC. El propósito es ofrecer a los usuarios una interfaz comprensible y accionable sobre los compromisos temporales del proyecto.

6.3.1 Extracción de fechas y plazos

A través de un sistema RAG con *prompting* jurídico especializado, se identificaron y clasificaron eventos temporales explícitos e implícitos presentes en el contrato.

Se generaron más de 30 eventos contractuales, incluyendo:

- Plazos obligatorios (**Deadlines**)
- Derechos del propietario (**Entitlements**)
- Hitos de ejecución (**Milestones**)

Tipo	Ejemplo extraído
`Deadline`	*"Performance Security Renewal" antes del 30/06/2024*
`Entitlement`	*"Change Order Due to Owner-Caused Delay" tras evento de demora*
`Milestone`	*"Site Preparation Completion and Owner Confirmation" el 30/06/2023*
`Other`	Eventos sin fecha pero relevantes para trazabilidad (ej.: formaciones)

Table 6.2: Tipos de eventos contractuales clasificados

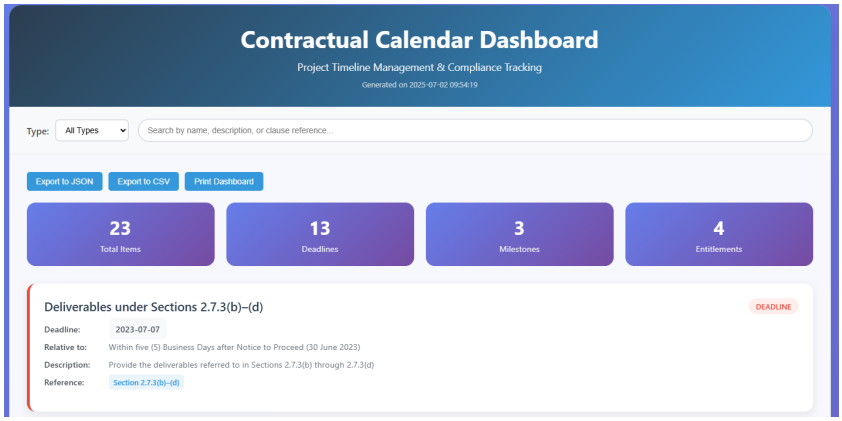


Figure 6.7: Vista General del Calendario Contractual

Cada evento incluye:

- Fecha absoluta (cuando está disponible)
- Referencia relativa (por ejemplo, *"5 días hábiles después del NTP"*)
- Descripción contextual
- Cláusula de referencia contractual

6.3.2 Clasificación de eventos

Cada entrada fue anotada con un campo `type` que permite discriminar entre distintas categorías legales. A continuación, se muestran ejemplos representativos:

6.3.3 Referencias y citas

Aunque muchas entradas no presentan aún citas literales (`source_citations`) debido a limitaciones técnicas actuales, el calendario incluye:

- Referencia a cláusulas (por ejemplo, `"Section 2.12.2"`)
- En algunos casos, fragmentos del contrato (`chunks`) que justifican la interpretación
- Indicación de fechas relativas basadas en eventos clave como el *Notice to Proceed* (NTP)

Esto permite trazabilidad legal aún en ausencia de fechas absolutas fijas.

6.3.4 Visualización interactiva

El calendario se exportó en formato HTML interactivo (`generated_calendar.html`), permitiendo:

- Filtros por tipo de evento (`Milestone`, `Deadline`, etc.)
- Orden cronológico o agrupación por dependencia
- Visualización optimizada para navegación legal y cumplimiento

Esto se podrá comprobar en los links adjuntos en el ANEXO B**??** que llevarán a las demostraciones de los productos.

6.3.5 Observaciones

Durante la evaluación se observaron los siguientes aspectos:

- Algunas entradas no fueron completamente decodificadas (error: `Double decoding failed`), lo que indica puntos de mejora en la robustez del *parser JSON*.
- Se incluyeron fechas clave del proyecto, como la entrega de garantías, emisión del NTP, fechas límite de cumplimiento, mecanismos de penalización y ventanas de rescisión anticipada.

6.3.6 Conclusiones

La generación automática del calendario contractual ha demostrado ser una herramienta de gran valor para:

- Centralizar y visualizar compromisos temporales dispersos en el contrato.
- Facilitar la supervisión del cumplimiento y preparación de hitos.
- Asistir en la ejecución del contrato por parte del equipo legal, técnico y de *compliance*.

Este componente es esencial para construir sistemas automatizados de *Contract Lifecycle Management* (CLM), especialmente en contratos EPC de alta complejidad.

6.4 Contratos EPC como caso de uso: complejidad y aplicabilidad en otros contextos

El contrato EPC (Engineering, Procurement and Construction) representa uno de los formatos contractuales más complejos dentro del ámbito industrial y energético, ya que concentra en un único documento obligaciones técnicas, plazos exigentes, hitos financieros, penalizaciones y cláusulas de resolución anticipada. Esta densidad legal y operativa lo convierte en un excelente caso de uso para probar la capacidad de un sistema inteligente de análisis contractual. Al abordar desde el inicio un contrato de esta envergadura, el sistema desarrollado ha demostrado ser capaz de manejar estructuras complejas, relaciones cruzadas y cláusulas sensibles, lo que sienta las bases para escalar su aplicación a todo tipo de contratos, desde acuerdos de confidencialidad y compraventa hasta contratos laborales, comerciales o de prestación de servicios.

Conclusión

7.1 Objetivos cumplidos

A lo largo del desarrollo de este Trabajo de Fin de Grado se han cumplido en su totalidad los objetivos inicialmente planteados, estructurados en dos grandes etapas: un primer módulo centrado en la anonimización de contratos conforme al RGPD, y un segundo módulo enfocado en el diseño de un sistema de gestión del ciclo de vida contractual (CLM), especializado en contratos EPC, mediante tecnologías de inteligencia artificial avanzada.

Durante la primera fase del proyecto se abordó con éxito la problemática de la protección de datos en documentos contractuales. Se desarrolló un sistema capaz de detectar y anonimizar de forma automática datos sensibles —como nombres propios, direcciones, cuentas bancarias o identificadores fiscales— utilizando modelos de lenguaje desplegados tanto en local (para entornos offline) como en la nube, haciendo uso de AWS Bedrock para garantizar privacidad y eficiencia. Este sistema fue integrado funcionalmente en un complemento (add-in) para Microsoft Word, lo que permite al usuario trabajar con sus documentos en un entorno familiar, sin comprometer la seguridad ni la operatividad.

Asimismo, se incorporó un mecanismo de verificación de entidades detectadas, de modo que el usuario puede revisar, confirmar o modificar los resultados antes de proceder a la edición o el intercambio del documento. Esta capa de validación refuerza la robustez del sistema y su alineación con el marco normativo vigente.

En la segunda fase del proyecto, se avanzó hacia un sistema mucho más ambicioso e innovador: la automatización del seguimiento contractual mediante inteligencia artificial. Para ello, se diseñó una arquitectura híbrida que combina técnicas de Retrieval-Augmented Generation (RAG) con la construcción de un grafo de conocimiento semántico, que permite interpretar, estructurar y visualizar el contenido jurídico de un contrato complejo.

Este sistema es capaz de responder consultas jurídicas específicas, generar even-

tos contractuales estructurados (como plazos, hitos o penalizaciones), verificar sus fundamentos en el texto original y representarlos en un calendario visual interactivo, especialmente adaptado a contratos de tipo EPC. La solución desarrollada va más allá de un simple extractor de fechas: es capaz de razonar, justificar y presentar la información de forma clara y accionable para los distintos perfiles involucrados en la gestión contractual.

En conjunto, puede afirmarse que todos los objetivos definidos al inicio del proyecto se han alcanzado de manera satisfactoria, y en muchos casos, superando las expectativas iniciales. La integración efectiva entre modelos de lenguaje, estructuras de grafos, procesamiento documental y visualización final ha dado lugar a un sistema completo, funcional y alineado con las necesidades reales del sector legal y de la ingeniería. Más allá de los logros técnicos, el desarrollo de este TFG ha sido una experiencia profundamente formativa, que ha permitido aplicar conocimientos teóricos a un problema real, explorar el potencial de las tecnologías emergentes y contribuir con una propuesta concreta al futuro del análisis jurídico automatizado.

7.2 Necesidad real

Este Trabajo de Fin de Grado no solo ha sido una experiencia académica de innovación técnica, sino también una propuesta con una clara vocación de aplicabilidad real. Tanto el sistema de anonimización como la arquitectura de gestión contractual desarrollada responden a necesidades concretas del mercado actual: el cumplimiento normativo en el tratamiento de datos sensibles y la automatización inteligente del análisis de contratos complejos, especialmente en sectores como el energético, la construcción o la ingeniería industrial.

La solución propuesta demuestra ser modular, escalable y orientada al usuario, lo que facilita su integración en entornos corporativos existentes y abre la puerta a una futura comercialización como producto legal-tech. La apuesta por tecnologías punteras como los modelos de lenguaje, la computación en la nube y los grafos de conocimiento posiciona este proyecto no solo como una prueba de concepto funcional, sino como una base sólida para el desarrollo de un producto competitivo, diferenciado y con alto potencial de impacto en el sector jurídico y de gestión de contratos EPC.

7.3 Proyecto de ingeniería

Este Trabajo de Fin de Grado se inscribe de forma natural en el ámbito de la ingeniería por múltiples razones que van desde la complejidad técnica del sistema

desarrollado hasta su orientación a la resolución estructurada de un problema real. En primer lugar, el proyecto aborda el diseño e implementación de una arquitectura completa basada en principios fundamentales de la ingeniería de software: modularidad, escalabilidad, eficiencia computacional, trazabilidad, interoperabilidad y seguridad.

El sistema combina de forma rigurosa y estructurada distintos bloques funcionales: procesamiento de lenguaje natural, anonimización de datos sensibles, recuperación semántica, generación estructurada de información, visualización interactiva y despliegue sobre infraestructura cloud. Esta integración requiere una planificación precisa, gestión de recursos, evaluación de rendimiento y control de errores, todos ellos aspectos clave de cualquier solución de ingeniería.

Además, el proyecto responde a una necesidad práctica y mensurable del sector —la automatización de procesos legales y el cumplimiento normativo— y propone una solución tecnológica con viabilidad técnica, impacto real y potencial de implementación industrial. El uso de herramientas como AWS, APIs de modelos fundacionales, técnicas de recuperación de información y grafos semánticos demuestra la aplicación efectiva de conocimientos avanzados adquiridos durante el grado.

Finalmente, el enfoque adoptado a lo largo del trabajo refleja el método ingenieril por excelencia: definición del problema, análisis de requisitos, diseño modular de la solución, implementación, pruebas, validación y documentación. Por todo ello, este TFG no solo cumple con los estándares académicos, sino que representa un ejercicio completo y profesional de ingeniería aplicada a un contexto emergente y de gran relevancia.

Líneas futuras

8.1 Evolución hacia una plataforma modular y profesional

El presente Trabajo de Fin de Grado sienta las bases de una solución avanzada de análisis y gestión contractual asistida por inteligencia artificial, pero también abre una hoja de ruta clara hacia su evolución como producto tecnológico completo. Las mejoras y extensiones previstas se dividen en torno a los dos módulos principales —la anonimización de documentos y el sistema de gestión del ciclo de vida contractual (CLM)—, convergiendo en una visión unificada de plataforma flexible y personalizable para entornos legales y documentales.

8.1.1 Evolución del módulo de anonimización

El primer bloque de mejoras se centra en transformar el sistema de anonimización desarrollado en una herramienta plenamente utilizable por profesionales. Para ello, se plantean las siguientes líneas de desarrollo:

- **Despliegue comercial del add-in para Microsoft Word**: se prevé la publicación del complemento en la tienda oficial de Microsoft Office, permitiendo a cualquier usuario descargar, instalar y utilizar el anonimizador directamente desde su entorno de trabajo habitual, sin necesidad de conocimientos técnicos ni configuración manual.
- **Ampliación multilingüe del sistema de detección**: se incorporarán nuevos modelos y reglas que permitan anonimizar documentos redactados en varios idiomas (especialmente inglés, francés y portugués), adaptándose así a contratos internacionales y ampliando el alcance comercial del sistema.
- **Integración de un asistente generativo mediante RAG**: como evolución del sistema puramente pasivo de anonimización, se plantea la incor-

poración de un asistente de redacción legal que, mediante una arquitectura RAG, permita al usuario generar borradores de cláusulas, adaptar textos y construir contratos nuevos a partir de ejemplos, todo ello desde el mismo entorno de Word.

Estas mejoras buscan no solo ampliar las capacidades del anonimizador, sino también posicionarlo como una herramienta activa y asistida para la creación, edición y compartición segura de documentación legal.

8.1.2 Evolución del módulo CLM

El módulo de gestión del ciclo de vida contractual también presenta un gran potencial de desarrollo, tanto en su dimensión técnica como en su aplicabilidad práctica. Las principales líneas futuras son:

- **Verificación interactiva de eventos dentro de Word**: una mejora clave es permitir que el usuario revise, valide o descarte manualmente los eventos generados por el sistema directamente desde la interfaz de Word, utilizando como apoyo las citas literales identificadas por el sistema. Esta verificación previa mejora la trazabilidad y adapta el calendario a las prioridades del usuario.
- **Rediseño de la visualización temporal**: se plantea una mejora de la interfaz de visualización mediante la creación de una línea temporal interactiva y filtrable, que facilite la comprensión cronológica del contrato y permita a distintos usuarios (legales, técnicos, de operaciones) consultar los hitos relevantes de forma clara.
- **Generalización del sistema a todo tipo de contratos**: actualmente centrado en contratos EPC, se busca adaptar el sistema para su aplicación a contratos de cualquier naturaleza, ampliando así el público objetivo y permitiendo que la plataforma funcione como un motor CLM adaptable.
- **Construcción de una plataforma SaaS comercializable**: el objetivo final es ofrecer un sistema completo y profesional donde los usuarios puedan cargar sus contratos, generar calendarios automáticamente y realizar el seguimiento de sus obligaciones y derechos contractuales desde un entorno centralizado.

8.1.3 Convergencia: hacia una plataforma modular y personalizable

Ambas líneas de desarrollo —anonimización y CLM— convergen en la visión de una plataforma SaaS flexible y modular, que permita a cada usuario personalizar

su entorno de trabajo jurídico a través de un único complemento para Microsoft Word. Esta plataforma permitiría activar o desactivar funcionalidades según las necesidades del usuario: anonimización, redacción asistida, verificación de eventos, visualización de calendario, entre otras.

El resultado sería un producto robusto, competitivo y adaptable, capaz de integrarse en los flujos de trabajo de despachos jurídicos, departamentos legales corporativos o empresas del sector industrial, y de proporcionar valor real mediante la automatización y estructuración del conocimiento contractual.

8.2 Perspectiva de futuro en el contexto de la evolución de la inteliencia artificial

El rápido avance de la inteligencia artificial, especialmente en el ámbito de los modelos de lenguaje y el procesamiento del lenguaje natural, dibuja un escenario extraordinariamente favorable para el perfeccionamiento continuo de sistemas como el desarrollado en este Trabajo de Fin de Grado. Cada nueva generación de modelos ofrece mejoras en comprensión semántica, capacidad de razonamiento y adaptación al contexto, lo que abre la puerta a soluciones cada vez más precisas, explicables y útiles.

En este sentido, el sistema de anonimización y gestión contractual aquí propuesto no debe entenderse como un producto cerrado, sino como una base sólida sobre la que crecer. A medida que los modelos se vuelvan más eficientes, multilingües y alineados con los objetivos del usuario, también lo hará la capacidad de estas herramientas para adaptarse a casos más complejos, reducir errores y ofrecer valor añadido en tiempo real.

Además, la evolución constante de las infraestructuras cloud, la aparición de modelos más ligeros y la democratización del acceso a tecnologías avanzadas permitirán que estas soluciones sean más accesibles, integrables y seguras. En conjunto, la innovación en inteligencia artificial no solo favorecerá el desarrollo de productos más potentes, sino que también contribuirá a redefinir la forma en que interactuamos con los documentos legales, transformando radicalmente los procesos de análisis, redacción y gestión contractual.

8.2.1 Proyección comercial y desarrollo de colaboraciones

Más allá del alcance técnico del proyecto, una de las líneas futuras con mayor potencial está ligada a su proyección comercial y de adopción temprana. El objetivo a corto y medio plazo es identificar y captar early adopters —profesionales del ámbito jurídico, despachos especializados, departamentos legales corporativos

o ingenierías con alta carga contractual— que estén interesados en integrar estas herramientas en sus flujos de trabajo.

A través de estos primeros casos de uso reales, se podrán validar las funcionalidades en entornos productivos, detectar necesidades específicas y construir relaciones de confianza que deriven en proyectos de consultoría tecnológica personalizada. Estas colaboraciones permitirán adaptar e implementar los sistemas desarrollados —tanto el anonimizador como el gestor de eventos contractuales— a contextos concretos, fortaleciendo su valor añadido y acelerando su maduración como producto.

Este enfoque no solo persigue una validación de mercado, sino también el crecimiento orgánico del proyecto como una solución a medida para organizaciones que buscan automatizar, estructurar y proteger sus procesos documentales mediante inteligencia artificial. La consultoría tecnológica, entendida como un puente entre la investigación y la implementación real, será una vía natural para convertir este trabajo académico en una herramienta transformadora y comercialmente viable.

Anexo A: Diagramas OPM con descripción OPL

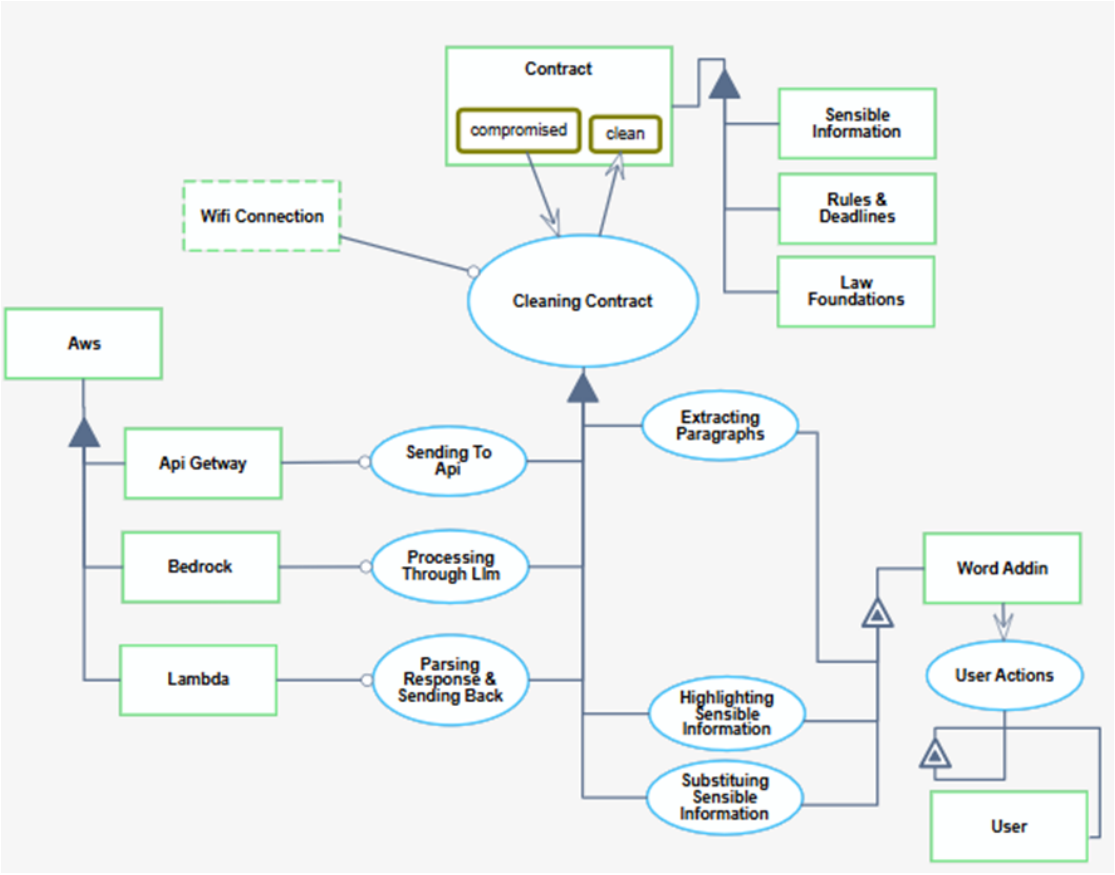


Figure 9.1: Diagrama OPM Arquitectura de anonimización

1. Cleaning Contract consists of Extracting Paragraphs, Highlighting Sensible Information, Parsing Response & Sending Back, Processing Through LLM, Sending To API, and Substituting Sensible Information.
2. Word Addin is an informatical and systemic object.
3. Word Addin exhibits Extracting Paragraphs, Highlighting Sensible Information, and Substituting Sensible Information.
4. User is an informatical and systemic object.
5. AWS is an informatical and systemic object.
6. API Gateway is an informatical and systemic object.
7. Bedrock is an informatical and systemic object.
8. Lambda is an informatical and systemic object.
9. AWS consists of API Gateway, Bedrock, and Lambda.
10. Contract is an informatical and systemic object.
11. Contract can be clean or compromised.
12. Sensible Information is an informatical and systemic object.
13. Rules & Deadlines is an informatical and systemic object.
14. Law Foundations is an informatical and systemic object.
15. Contract consists of Law Foundations, Rules & Deadlines, and Sensible Information.
16. WiFi Connection is an informatical and environmental object.
17. User exhibits User Actions.
18. Cleaning Contract is an informatical and systemic process.
19. Cleaning Contract changes Contract from compromised to clean.
20. Cleaning Contract requires WiFi Connection.
21. Extracting Paragraphs of Word Addin is an informatical and systemic process.

22. Sending To API is an informatical and systemic process.
23. Sending To API requires API Gateway.
24. Processing Through LLM is an informatical and systemic process.
25. Processing Through LLM requires Bedrock.
26. Parsing Response & Sending Back is an informatical and systemic process.
27. Parsing Response & Sending Back requires Lambda.
28. Highlighting Sensible Information of Word Addin is an informatical and systemic process.
29. Substituting Sensible Information of Word Addin is an informatical and systemic process.
30. User Actions of User is an informatical and systemic process.
31. User Actions of User consumes Word Addin.

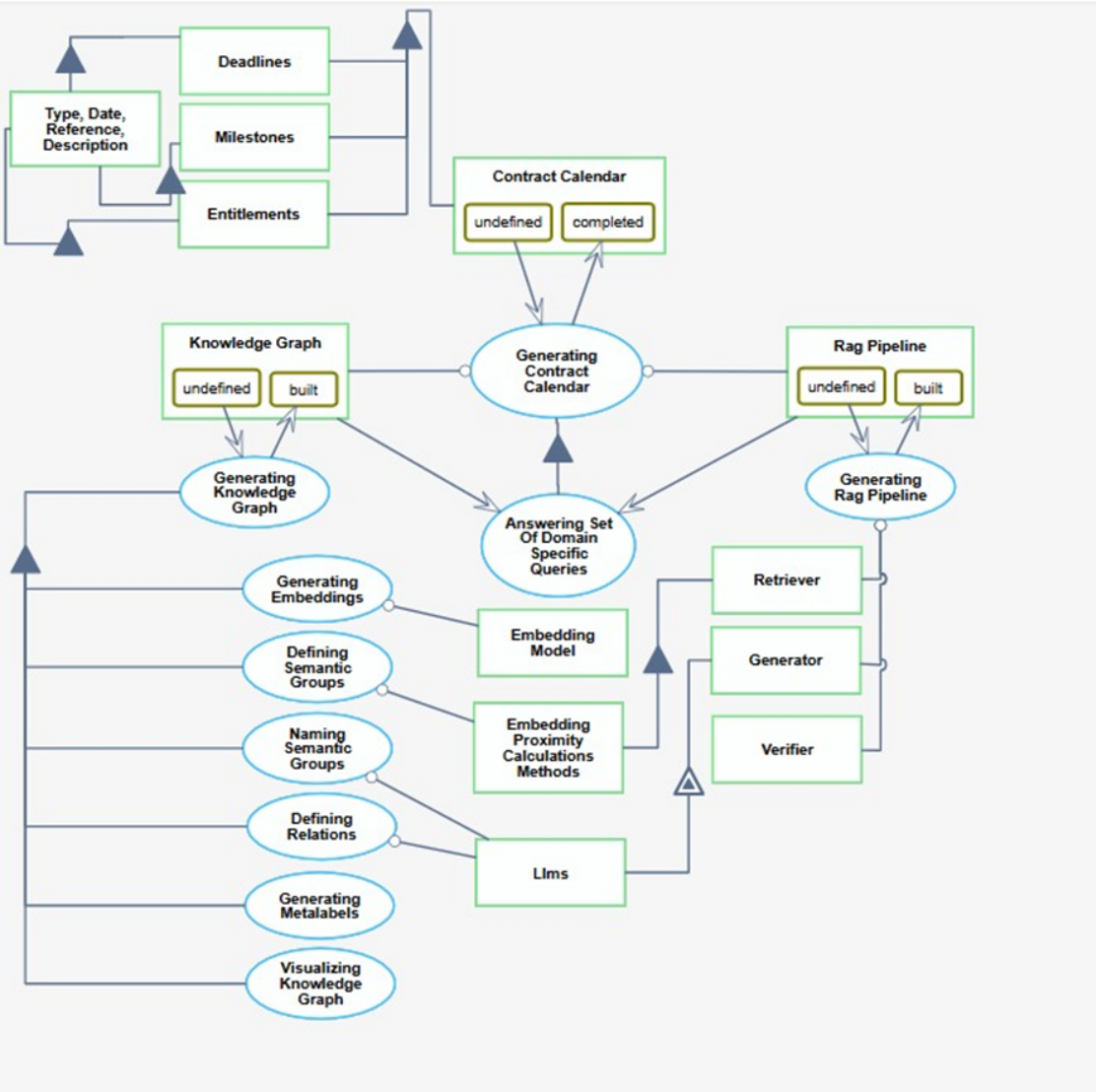


Figure 9.2: Diagrama OPM Arquitectura CLM

1. Contract Calendar is an informatical and systemic object.
2. Contract Calendar can be completed or undefined.
3. Rag Pipeline is an informatical and systemic object.
4. Rag Pipeline can be built or undefined.
5. Knowledge Graph is an informatical and systemic object.
6. Knowledge Graph can be built or undefined.
7. Generating Knowledge Graph consists of Defining Relations, Defining Semantic Groups, Generating Embeddings, Generating Metalabels, Naming Semantic Groups, and Visualizing Knowledge Graph.
8. LLMs of Generator is an informatical and systemic object.
9. Embedding Model is an informatical and systemic object.
10. Embedding Proximity Calculations Methods is an informatical and systemic object.
11. Deadlines is an informatical and systemic object.
12. Milestones is an informatical and systemic object.
13. Entitlements is an informatical and systemic object.
14. Contract Calendar consists of Deadlines, Entitlements, and Milestones.
15. Type, Date, Reference, Description is an informatical and systemic object.
16. Deadlines consists of Type, Date, Reference, Description.
17. Entitlements consists of Type, Date, Reference, Description.
18. Milestones consists of Type, Date, Reference, Description.
19. Retriever is an informatical and systemic object.
20. Generator is an informatical and systemic object.
21. Verifier is an informatical and systemic object.
22. Generator exhibits LLMs.
23. Retriever consists of Embedding Proximity Calculations Methods.

24. Generating Contract Calendar consists of Answering Set Of Domain Specific Queries.
25. Generating Contract Calendar is an informatical and systemic process.
26. Generating Contract Calendar changes Contract Calendar from undefined to completed.
27. Generating Contract Calendar requires Knowledge Graph and Rag Pipeline.
28. Generating Rag Pipeline is an informatical and systemic process.
29. Generating Rag Pipeline changes Rag Pipeline from undefined to built.
30. Generating Rag Pipeline requires Generator, Retriever, and Verifier.
31. Generating Knowledge Graph is an informatical and systemic process.
32. Generating Knowledge Graph changes Knowledge Graph from undefined to built.
33. Generating Embeddings is an informatical and systemic process.
34. Generating Embeddings requires Embedding Model.
35. Defining Semantic Groups is an informatical and systemic process.
36. Defining Semantic Groups requires Embedding Proximity Calculations Methods.
37. Naming Semantic Groups is an informatical and systemic process.
38. Naming Semantic Groups requires LLMs of Generator.
39. Defining Relations is an informatical and systemic process.
40. Defining Relations requires LLMs of Generator.
41. Generating Metalabels is an informatical and systemic process.
42. Visualizing Knowledge Graph is an informatical and systemic process.
43. Answering Set Of Domain Specific Queries is an informatical and systemic process.
44. Answering Set Of Domain Specific Queries consumes Knowledge Graph and Rag Pipeline.

Anexo B: Links a repositorios y demos

- A continuación, el link al repositorio de Github que contiene los códigos tanto del CLM como del anonimizador:
  https://github.com/joseguardo/ClaudIA
- A continuación el link a los videos de YouTube que contienen demos de los módulos:
  Anonimizador: https://www.youtube.com/watch?v=Wl232RApeL8
  Chatbot: https://www.youtube.com/watch?v=ozNCdK0f2_c
  CLM: https://www.youtube.com/watch?v=z-T39tRiMro

Bibliography

[1] Abogacía Española. Microsoft Office 365 para abogados: 5 procesos fáciles para implementar desde ya en tu despacho o asesoría legal, 2022. Blog de Innovación Legal [online].

[2] Amazon Web Services, Inc. Amazon API Gateway Developer Guide, 2025. Accedido el 29 de junio de 2025.

[3] Amazon Web Services, Inc. *Amazon Bedrock: User Guide*. Amazon Web Services, 2025. Version retrieved from official documentation.

[4] Amazon Web Services, Inc. Amazon Bedrock – Precios, 2025. Accedido el 29 de junio de 2025.

[5] Amazon Web Services, Inc. Amazon Bedrock – Security and Privacy, 2025. Accedido el 29 de junio de 2025.

[6] Amazon Web Services, Inc. AWS Lambda – Computación serverless en la nube, 2025. Accedido el 29 de junio de 2025.

[7] Amazon Web Services, Inc. Boto3 documentation (AWS SDK for Python), 2025. Accedido el 29 de junio de 2025.

[8] Amazon Web Services, Inc. Conformidad con la norma ISO/IEC27001:2022 – Preguntas frecuentes, 2025. Accedido el 29 de junio de 2025.

[9] Amazon Web Services, Inc. Define Lambda function handler in Python – AWS Lambda Developer Guide, 2025. Accedido el 29 de junio de 2025.

[10] Amazon Web Services, Inc. Reglamento General de Protección de Datos (GDPR) – Centro de cumplimiento de AWS, 2025. Accedido el 29 de junio de 2025.

[11] Amazon Web Services, Inc. Use layers to manage your deployment package in AWS Lambda – AWS Lambda Developer Guide, 2025. Accedido el 29 de junio de 2025.

[12] Artificial Analysis. Tiny Open Source Models (<4B parameters) – Leaderboard and comparison, 2025. Accedido el 29 de junio de 2025.

[13] ARX Developers. ARX – Data Anonymization Tool: Software de anonimización de datos personales. `https://arx.deidentifier.org/`, 2025. Accedido el 27 de junio de 2025.

[14] Maryam Ashoori, Chintan Patel, and NVIDIA. IBM's New Granite 3.0 Generative AI Models Are Small, Yet Highly Accurate and Efficient, 2024. Accedido el 27 de junio de 2025.

[15] Ganesh Bajaj. Named Entity Recognition (NER) for sanitizing the PII and sensitive data for public LLMs, 2025. Accedido el 27 de junio de 2025.

[16] BIGLE Iberia S.L. Bigle Legal – Plataforma de gestión de contratos (Contract Lifecycle Management). `https://www.biglelegal.com/`, 2025. Accedido el 27 de junio de 2025.

[17] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., Sebastopol, CA, 2009.

[18] Boudribila. AutoFactory: dataset repository – README.md, 2025. Accedido el 28 de junio de 2025.

[19] Broadcom Inc. Test Data Manager – Solución para gestión de datos de prueba (antes CA Test Data Manager). `https://www.broadcom.com/products/software/app-dev/test-data-manager`, 2025. Accedido el 27 de junio de 2025.

[20] Frederick P. Brooks. The computer scientist as toolsmith ii. *Communications of the ACM*, 39(3):61–68, 1996.

[21] DocuSign, Inc. DocuSign CLM – Software de gestión del ciclo de vida de los contratos. `https://www.docusign.com/es-es/productos/clm`, 2025. Accedido el 27 de junio de 2025.

[22] Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, Jie Meng, Wenjie Ruan, and Xiaowei Huang. Building guardrails for large language models, 2024.

[23] Editores, Tecnobitt. LexDoka se consolida como una de las plataformas de gestión de contratos más innovadoras de 2025, 2025. Blog Tecnobitt [online].

[24] European Commission. Legal framework of EU data protection, 2025. Accedido el 29 de junio de 2025.

[25] Evisort, Inc. Evisort – Plataforma de gestión del ciclo de vida de contratos con IA. https://www.evisort.com/, 2025. Accedido el 27 de junio de 2025.

[26] Gutowska, Anna and IBM. Agentic RAG: Build a LangChain agentic RAG system using Granite-3.0-8B-Instruct in watsonx.ai, 2025. Accedido el 27 de junio de 2025.

[27] Ciaran M. Harper and S. Sarah Zhang. Legal tech and lawtech: Towards a framework for technological trends in the legal services industry. In Henner Gimpel, Rainer Alt, Dennis Kundisch, Philipp Kreuzer, and Matthias Schumann, editors, *Market Engineering: Insights from Two Decades of Research on Markets and Information Systems*, pages 183–206. Springer International Publishing, Cham, 2021.

[28] Icertis. Icertis – Contract Intelligence Platform. https://www.icertis.com/, 2025. Accedido el 27 de junio de 2025.

[29] Ironclad, Inc. Ironclad – Plataforma de gestión del ciclo de vida de contratos con IA. https://ironcladapp.com/, 2025. Accedido el 27 de junio de 2025.

[30] K2View. Data Masking Tools – Convencional + con IA, 2025. Accedido el 27 de junio de 2025.

[31] LangChain, Inc. LangChain – Framework for building applications with LLMs. https://www.langchain.com/, 2025. Accedido el 27 de junio de 2025.

[32] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[33] LexDoka S.L. LexDoka – Plataforma de gestión de contratos (Contract Lifecycle Management). https://lexdoka.com/, 2025. Accedido el 27 de junio de 2025.

[34] Microsoft Corporation. Office Add-ins with the add-in only manifest, 2024. Accedido el 30 de junio de 2025.

[35] Microsoft Corporation. Build your first Word task pane add-in – Quickstart (Yo Office), 2025. Accedido el 30 de junio de 2025.

[36] Microsoft Corporation. Office Addins platform overview, 2025. Accedido el 30 de junio de 2025.

[37] Mistral AI. Function Calling – Mistral AI Documentation, 2025. Accedido el 27 de junio de 2025.

[38] Mistral AI. Mistral AI Documentation, 2025. Accedido el 27 de junio de 2025.

[39] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):1–20, 2007. Accedido el 27 de junio de 2025.

[40] Ollama Inc. Ollama – Ejecuta modelos LLM localmente (DeepSeek-R1, Llama3.3, Qwen3, Gemma3, etc.). `https://ollama.com/`, 2025. Accedido el 29 de junio de 2025.

[41] OpenAI. Function calling guide – OpenAI Platform Docs, 2025. Accedido el 27 de junio de 2025.

[42] OpenAI. Tools – OpenAI Agents SDK (Python), 2025. Accedido el 27 de junio de 2025.

[43] Pallets Project. Jinja Documentation (stable), 2025. Accedido el 29 de junio de 2025.

[44] Python Software Foundation. `concurrent.futures` – Launching parallel tasks, 2025. Accedido el 29 de junio de 2025.

[45] Afsaneh Shams Sakher Khalil Alqaaidi, Elika Bozorgi and Krzysztof J. Kochut. Afew-shot learning-focused survey on recent named entity recognition and relation classification models. `https://www.scitepress.org/Papers/2024/127916/127916.pdf`, 2024. Accedido el 28 de junio de 2025.

[46] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023.

[47] Erik Schluntz, Barry Zhang, and Anthropic. Building Effective Agents – Anthropic Engineering Blog, 2024. Accedido el 27 de junio de 2025.

[48] SirionLabs Inc. Sirion.ai – Plataforma de gestión del ciclo de vida de contratos con IA. `https://www.sirion.ai/`, 2025. Accedido el 27 de junio de 2025.

[49] Syntho. Smart De-Identification – Plataforma de desidentificación inteligente de datos, 2025. Accedido el 27 de junio de 2025.

[50] Anh Truong, Austin Walters, and Jeremy Goodsitt. Sensitive data detection with high-throughput neural network models for financial institutions, 2020.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.