Anna Aragoneses Romero

IMPLEMENTACIÓ DE RAG MITJANÇANT GRAFS PER A LA MILLORA DE LA RECUPERACIÓ D'INFORMACIÓ

TREBALL DE FI DE GRAU

dirigit per Maria Ferré Bergadà

Grau d'Enginyeria Informàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2025

Resum.

Aquest Treball de Fi de Grau presenta la implementació d'un sistema de *Retrieval-Augmented Generation* (RAG) basat en graf de coneixement per millorar la recuperació i generació d'informació a partir de documents no estructurats. La motivació principal és donar resposta a la necessitat creixent d'aprofitar documents textuals, com ara PDFs tècnics, per extreure'n coneixement útil i explotable de manera automàtica.

La metodologia es basa en un procés en diverses etapes: primer, es realitza l'extracció d'entitats, atributs i relacions mitjançant eines com Amazon Comprehend i spaCy. A continuació, aquest coneixement s'estructura en un graf jeràrquic en Neo4j, amb diferents nivells de granularitat (document, pàgina, sentència, entitat, etc.). Paral·lelament, s'implementa un sistema de verificació automàtica mitjançant un agent LLM per assegurar la qualitat del coneixement extret. Finalment, s'integra un mòdul de RAG per respondre consultes de manera precisa utilitzant tant la base textual com el coneixement estructurat.

Els resultats obtinguts demostren que la integració d'un graf millora la interpretabilitat i la rellevància de les respostes generades pel sistema.

Resumen.

Este Trabajo de Fin de Grado presenta la implementación de un sistema de Retrieval-Augmented Generation (RAG) basado en grafos de conocimiento para mejorar la recuperación y generación de información a partir de documentos no estructurados. La motivación principal es responder a la creciente necesidad de aprovechar documentos textuales, como archivos PDF técnicos, para extraer conocimiento útil de forma automática.

La metodología seguida incluye varias fases: en primer lugar, se realiza la extracción de entidades, atributos y relaciones mediante herramientas como Amazon Comprehend y spaCy. Luego, este conocimiento se estructura en un grafo jerárquico en Neo4j con distintos niveles de granularidad (documento, página, sentencia, entidad, etc.). Paralelamente, se implementa un sistema de verificación automática mediante un agente LLM para garantizar la calidad del conocimiento extraído. Finalmente, se integra un módulo de RAG que permite responder preguntas de forma precisa utilizando tanto la base textual como el conocimiento estructurado.

Los resultados muestran que la integración de un grafo mejora la interpretabilidad y la relevancia de las respuestas generadas.

Abstract.

This Final Degree Project presents the implementation of a *Retrieval-Augmented Generation* (RAG) system based on knowledge graphs to improve information retrieval and generation from unstructured documents. The main motivation is to address the increasing need to leverage textual documents, such as technical PDFs, to automatically extract useful and actionable knowledge.

The methodology consists of several stages: first, entity, attribute, and relation extraction is performed using tools such as Amazon Comprehend and spaCy. Then, the extracted knowledge is structured into a hierarchical graph in Neo4j with multiple granularity levels (document, page, sentence, entity, etc.). In parallel, an automated verification system is implemented using an LLM agent to ensure the quality of the extracted knowledge. Finally, a RAG module is integrated to answer user queries accurately, combining both the textual base and the structured knowledge.

The results show that integrating a knowledge graph enhances the interpretability and relevance of generated answers.

Índex

Índex de figures

1 Introducció

En l'entorn digital actual, les organitzacions generen cada cop més informació en formats no estructurats, com **documents tècnics en PDF**, **informes textuals** o **correus electrònics [1]**. El repte principal és transformar aquest volum de text dispers en **coneixement útil i consultable**.

Els sistemes de **Recuperació Augmentada amb Generació (RAG, *Retrieval-Augmented Generation*)** han guanyat rellevància [2] perquè combinen models de llenguatge amb mecanismes de recuperació de context. Això permet obtenir respostes més precises i basades en evidències documentals.

Aquest treball sorgeix de la **proposta** d'Applus+ IDIADA, que actualment està desenvolupant un projecte relacionat amb la recuperació d'informació mitjançant **vectors** com a representació principal del coneixement. Tot i que el projecte industrial se centra en l'ús d'embeddings vectorials per a la cerca semàntica, el TFG planteja una aproximació alternativa basada **en grafs de coneixement**.

Tot i els avenços, un problema clau dels sistemes RAG és la **interpretabilitat [3]**. Les respostes poden ser útils, però sovint no queda clar d'on prové la informació ni si és fiable. Aquesta manca de transparència limita l'ús en àmbits crítics, com el **sector industrial**, **jurídic** o **sanitari**, on la traçabilitat és indispensable. Aquest **Treball de Final de Grau (TFG**), desenvolupat en **Applus+ IDIADA**, proposa un sistema RAG basat en **grafs de coneixement**. El graf permet representar de manera explícita **entitats**, **atributs** i **relacions** detectats en els documents, oferint una base estructurada per a consultes complexes i millorant la transparència.

El projecte també incorpora un **agent de verificació** basat en **models de llenguatge (LLM, *Large Language Models*)**, capaç d'analitzar el text original i validar la informació extreta. Això redueix errors i reforça la confiança en el sistema. El resultat és un enfocament híbrid —recuperació, generació i verificació— que mostra el valor d'integrar grafs i LLM per gestionar informació no estructurada.

Aquest plantejament té aplicacions en **assistents intel·ligents**, **motors de cerca semàntics** i **eines de suport a la decisió**. Alhora, contribueix a la recerca aplicada en **intel·ligència artificial (IA),** posant l'accent en sistemes més robustos, adaptatius i interpretables.

1.1 Objectius del projecte

L'objectiu general és desenvolupar una **prova de concepte** d'un sistema RAG que utilitzi **grafs de coneixement** com a estructura central. Es vol avaluar la seva utilitat en la **recuperació** i **verificació** d'informació provinent de documents no estructurats.

Objectius específics:

1. **Dissenyar i implementar l'arquitectura del sistema**
    - Definir una estructura modular amb fases clares: extracció d'informació, validació semàntica, construcció del graf i consulta.

- Garantir escalabilitat i possibilitat de substituir o millorar components de forma independent.

2. **Aplicar tècniques d'extracció d'informació consolidades**
   - Utilitzar **Amazon Comprehend** i **spaCy** per identificar entitats, relacions i atributs.
   - Organitzar els resultats de manera jeràrquica: document → pàgines → sentències → entitats → relacions.

3. **Incorporar un agent de verificació basat en LLM**
   - Desenvolupar un mòdul que validi automàticament entitats, relacions i atributs.
   - Generar informes de qualitat per detectar errors i inconsistències.

4. **Construir i gestionar el graf de coneixement**
   - Emprar **Neo4j** per a l'emmagatzematge i navegació del graf.
   - Definir esquemes de consultes per explorar les connexions semàntiques.

5. **Avaluar el sistema en un entorn experimental**
   - Testar la prova de concepte amb documents reals d'**Applus+ IDIADA**.
   - Formular preguntes específiques per mesurar precisió, traçabilitat i interpretabilitat.

6. **Analitzar la viabilitat i establir línies de futur**
   - Valorar la transferibilitat del sistema a altres dominis.
   - Identificar millores i possibles aplicacions industrials o acadèmiques.

2 Paraules clau

Els termes que millor descriuen el projecte desenvolupat:

- Bases de dades
- Programació
- Intel·ligència Artificial
- Machine learning

3 Descripció general del projecte

3.1 Context

En l'era digital, les organitzacions acumulen grans volums de **documentació tècnica**, **informes** i **dades en formats no estructurats**, sobretot en **PDF** o text lliure. Aquest contingut és difícil de processar automàticament perquè no segueix cap esquema fix.

El projecte parteix d'aquest repte i té com a objectiu transformar coneixement no estructurat en **representacions formals i navegables** mitjançant **grafs semàntics**. El desenvolupament es fa en col·laboració amb **Applus+ IDIADA**, empresa d'enginyeria de l'automoció, tot i que el sistema és agnòstic respecte al domini i es pot adaptar a altres àmbits.

3.2 Motivació

La motivació principal és millorar l'accés i l'ús del coneixement present en documents textuals. Els sistemes de preguntes i respostes basats en **models de llenguatge (LLM, *Large Language Models*)** acostumen a treballar amb text complet. Això pot generar respostes poc interpretables o amb fonament incert.

El projecte proposa una alternativa estructurada i verificable: transformar el text en un **graf de coneixement jeràrquic**, enriquit amb una **capa de validació automàtica** que comprova la coherència de la informació. D'aquesta manera, és possible formular preguntes específiques i obtenir respostes **traçables i justificades**.

3.3 Tasques generals

Les tasques principals del projecte són:

- Analitzar els requisits generals per a un sistema **RAG (Retrieval-Augmented Generation)** basat en graf.
- Dissenyar una arquitectura modular dividida en fases independents.
- Implementar components per a l'extracció d'informació semàntica (entitats, relacions i atributs) a partir de documents PDF.
- Construir un **graf de coneixement** en **Neo4j**, seguint una jerarquia estructurada.
- Desenvolupar un agent de validació basat en LLM per verificar i corregir les extraccions.
- Estudiar la recuperació d'informació mitjançant preguntes sobre el graf i analitzar les respostes generades.
- Documentar i avaluar els resultats obtinguts.

3.4 Entorn

El desenvolupament s'ha dut a terme en un **entorn de recerca**, combinant tecnologies locals i serveis al núvol.

- **Extracció semàntica:** serveis com **Amazon Comprehend** i la llibreria **spaCy**.
- **Gestió del graf:** base de dades **Neo4j**.

- **Validació**: un agent de llenguatge LLM integrat amb la plataforma **Amazon Bedrock**.

L'entorn inclou scripts de processament i una estructura de carpetes i fitxers orientada al tractament per pàgina de cada document. També s'han utilitzat eines d'anàlisi i visualització per verificar i demostrar els resultats.

3.5 Necessitats

La implementació ha requerit coneixements en **NLP (*Natural Language Processing*)**, modelatge de grafs i ús d'**APIs de serveis cognitius**. També ha estat necessària una infraestructura mínima capaç de processar múltiples documents i gestionar dades complexes. A més, ha calgut disposar d'un conjunt suficient de documents per a la prova de concepte i accés a un model de llenguatge potent i fiable per a la validació, mitjançant la integració amb LLM comercials via **AWS (Amazon Web Services)**. En l'àmbit organitzatiu, ha estat clau mantenir una comunicació fluida amb l'empresa col·laboradora per entendre el tipus de documents i els requisits del domini real.

3.6 Previsions d'ús

El projecte es presenta com una **prova de concepte**, però el seu disseny modular i agnòstic facilita l'extensió a diversos contextos. En un futur, es podria adaptar a sectors com la **medicina**, la **indústria farmacèutica**, el **dret** o l'**administració pública**, on la informació crítica sovint es troba en documents no estructurats. Els usos potencials inclouen:

- Sistemes de suport a la decisió.
- Assistents intel·ligents per a consultes tècniques.
- Automatització de processos de revisió documental.
- Extracció d'indicadors a partir de normatives tècniques.

3.7 Problemes i solucions

Durant el projecte es van detectar problemes conceptuals i tècnics que podien afectar la qualitat del coneixement extret i la funcionalitat del sistema. A continuació es descriuen els principals reptes i les solucions adoptades.

Problema 1. Extracció incompleta o incorrecta d'informació.

Els **PDF** presentaven dificultats: alguns eren escanejats i d'altres tenien formats irregulars. Això provocava que entitats i relacions importants no fossin detectades, reduint la precisió del graf.

Solució. Es va implementar un **pipeline híbrid** combinant **OCR (Optical Character Recognition)** per als documents escanejats i eines de processament de text estructurat com **Amazon Comprehend** i **spaCy**. Per exemple, en una normativa tècnica, l'OCR permetia llegir taules amb dades de costos, mentre que Comprehend identificava entitats com *Vehicle*, *Component* i *Requisit de seguretat*. Amb aquesta combinació, la cobertura d'extracció va arribar al **95%** en proves preliminars.

Problema 2. Conflictes i incoherències en el graf de coneixement.

En alguns casos hi havia entitats duplicades o atributs contradictoris, cosa que generava inconsistències i dificultava la recuperació fiable de context per al model **RAG (Retrieval-Augmented Generation).**

Solució. Es va afegir una etapa de verificació i normalització. Un **agent LLM (Large Language Model)** revisava entitats i relacions per detectar duplicats i incoherències. Per exemple, si apareixia la mateixa peça amb dues denominacions (*Sensor de velocitat* i *Speed sensor*), el sistema unificava l'entitat i mantenia tots els atributs associats. Així es garantí la coherència semàntica del graf.

Problema 3. Gestió de consultes complexes.

El sistema havia de respondre preguntes específiques sobre documents reguladors amb múltiples apartats i annexos. Les consultes senzilles funcionaven, però les que combinaven informació de diverses pàgines eren difícils de processar.

Solució. Es va adoptar un mecanisme de **retrieval adaptatiu per finestres de context**, segmentant els documents en pàgines i recuperant només els fragments rellevants [4]. Per exemple, davant la pregunta *"Quins requisits de seguretat s'apliquen al component X en condicions extremes?"*, el sistema identificava les pàgines pertinents i les enviava al model RAG, evitant saturar la memòria del model i millorant la precisió.

Problema 4. Escalabilitat i sostenibilitat.

Amb grans volums de documents, el processament i la construcció del graf es tornaven lents i difícils de mantenir.

Solució. Es va dissenyar un **pipeline modular i paral·lelitzat**, amb processos independents per a l'extracció, la verificació i la importació al graf [5].

Problema 5. Consultes amb formats especials (notes de peu o símbols).

Algunes preguntes contenien elements no convencionals, com superíndexs o referències a notes de peu (p. ex. *"Com medir la resistència U[1]*"). Aquestes entrades no es reconeixien correctament i quedaven sense correspondència al graf.

Solució (pendent). No s'ha implementat encara, però es planteja afegir un **preprocessador de consultes** per netejar i normalitzar l'entrada abans de passar-la al mòdul de recuperació [6]. Una altra opció seria un mòdul de *query rewriting* que reformuli automàticament la pregunta en un format compatible.

Problema 6. Limitacions de tokens i saturació en proves massives.

En proves amb centenars de preguntes seqüencials, van aparèixer errors pels límits de tokens establerts a **AWS (Amazon Web Services) Bedrock**.

L'error típic era:

ERROR │ Error al procesar la pregunta '¿Qué necesidad se reconoce?': An error occurred (ThrottlingException) when calling the InvokeModel operation (reached max retries: 4): Too many tokens, please wait before trying again.

Solució (pendent).

Una possible estratègia és implementar un sistema de **gestió de cues** amb *rate limiting* i control del nombre màxim de tokens per crida. També es podria aplicar **batching dinàmic**, agrupant preguntes petites en una sola crida, i explorar l'ús de **resums incrementals** per reduir la càrrega de tokens en proves massives.

4 Fonaments teòrics

4.1 Grafs de coneixement

Un **graf de coneixement** és una estructura de dades que representa informació mitjançant **nodes** i **relacions [7]**. Cada node correspon a una entitat, concepte o objecte, i cada aresta indica una connexió o associació entre ells.

Aquest model permet integrar dades de fonts diferents i oferir-ne una visió semàntica i interconnectada. A més d'emmagatzemar informació, facilita la seva consulta i el raonament automàtic a través de tècniques de recuperació de coneixement, inferència i aprenentatge automàtic (*machine learning*).

Característiques principals

- **Flexibilitat i extensibilitat**: els grafs poden créixer amb nous nodes i relacions sense haver de modificar una estructura rígida, a diferència d'una base de dades relacional (BDR).
- **Expressivitat semàntica**: nodes i relacions poden portar etiquetes i propietats que n'enriqueixen el context.
- **Integració de dades**: permeten combinar informació de diferents orígens i millorar la interoperabilitat entre sistemes.

Aplicacions habituals

- S**istemes de recomanació**: connecten usuaris, productes i preferències.
- **Recerca d'informació avançada:** milloren la precisió contextualitzant consultes i resultats.
- **Assistents intel·ligents i *chatbots***: permeten accedir a coneixement estructurat per respondre preguntes complexes.
- **Anàlisi i visualització**: revelen patrons i relacions dins grans volums de dades.

En el projecte

El graf de coneixement s'utilitza com a base de dades semàntiques on s'emmagatzemen **entitats**, **atributs** i **relacions** extrets de documents PDF. Aquesta estructura dona suport a una recuperació adaptativa i contextualitzada de la informació per respondre consultes en llenguatge natural.

S'ha fet servir **Neo4j**, una base de dades orientada a grafs, que permet representar i consultar informació amb el llenguatge **Cypher [8]**. Aquest enfocament facilita explotar les connexions entre entitats i obtenir respostes més precises.

4.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) és una tècnica del **Processament del Llenguatge Natural (PLN, *Natural Language Processing*)** que combina dues capacitats clau dels **models de llenguatge grans (LLM, *Large Language Models*)**: **recuperació d'informació** i **generació de text [9]**.

Aquest enfocament integra coneixement extern de fonts estructurades o no estructurades (documents, bases de dades, etc.) amb la potència generativa dels LLMs, millorant la precisió i la rellevància de les respostes.

Principis bàsics

Els models de llenguatge convencionals tenen limitacions per recordar informació molt específica o actualitzada, ja que només coneixen les dades amb què han estat entrenats.

RAG supera aquest límit amb un **component de recuperació** que localitza documents o fragments rellevants i un **component generatiu** que utilitza aquesta informació per produir respostes coherents i contextualitzades.

Els dos passos bàsics són:

1. **Recuperació (Retrieval):** A partir d'una consulta, el sistema localitza documents o fragments rellevants. S'utilitzen motors de cerca, **embeddings semàntics** o consultes a bases de dades de graf.

2. **Generació (Generation):** El model generatiu (LLM) produeix la resposta usant el context recuperat, adaptant-se a la pregunta de l'usuari.

Arquitectura típica de RAG

Un model RAG té tres components principals [10]:

- **Índex o base de coneixement:** Documents, fragments de text o nodes d'un graf de coneixement, organitzats per permetre recuperacions ràpides i rellevants.
- **Mòdul de recuperació:** Analitza la consulta i identifica els fragments més pertinents. Sovint utilitza embeddings semàntics i cerca vectorial.
- **Model generatiu:** Un LLM que genera la resposta a partir de la informació recuperada. Pot ser, per exemple, **GPT**, **Claude** o qualsevol altre LLM capaç d'integrar context addicional.

Avantatges de RAG

- **Actualització dinàmica del coneixement:** Permet usar informació nova sense reentrenar el model.
- **Precisió i rellevància:** Redueix respostes imprecises o inventades (*hallucinations*), ja que es basa en documents reals.
- **Escalabilitat:** Funciona amb grans volums de dades simplement actualitzant l'índex de recuperació.

Aplicació de RAG en el projecte

El sistema aplica RAG per respondre preguntes sobre documents tècnics (PDFs) de l'empresa:

- El contingut dels documents s'emmagatzema en un **graf de coneixement**.
- Quan arriba una consulta, es recuperen nodes o fragments rellevants del graf.

- Aquests fragments s'envien al LLM (**Claude 3.5 Sonnet a AWS Bedrock**) que genera la resposta final contextualitzada [11].

Aquesta combinació millora la qualitat, la fiabilitat i la contextualització de les respostes respecte a un model generatiu pur.

Limitacions i reptes

- **Qualitat de la recuperació:** La precisió depèn dels documents localitzats.
- **Integració i sincronització:** Cal assegurar que la informació recuperada s'integra correctament per evitar incoherències.
- **Cost computacional:** La combinació de recuperació i generació pot ser exigent en recursos i temps de processament.

4.3 Processament del llenguatge natural (PLN)

El **Processament del Llenguatge Natural (PLN, *Natural Language Processing*)** és una branca de la **intel·ligència artificial** i la **lingüística computacional [12]**. L'objectiu principal és que els ordinadors entenguin, interpretin, generin i manipulin el llenguatge humà de manera útil i natural.

El PLN transforma textos o discursos en dades estructurades, permet interpretar significats, respondre preguntes, traduir idiomes, resumir textos, detectar sentiments i molt més.

Principals components i processos

El PLN inclou diversos nivells d'anàlisi [13]:

- **Anàlisi morfològica**: Divideix les paraules en unitats més petites (*morfemes*) i assigna categories gramaticals (nom, verb, adjectiu...).
- **Anàlisi sintàctica:** Estudia l'estructura de les frases per identificar relacions gramaticals (subjecte, verb, objecte). S'usen arbres sintàctics i anàlisi de dependències.
- **Anàlisi semàntica:** Assigna significat a paraules i frases, detectant entitats, relacions i intencions.
- **Anàlisi pragmàtica:** Interpreta el context i la intenció de l'emissor per comprendre significats implícits, ironies o ambigüitats.
- **Reconeixement d'entitats i extracció de relacions (NER i RE):** Identifica entitats com persones, llocs i organitzacions, i les connecta amb relacions per crear representacions estructurades, com **grafs de coneixement**.

A**plicacions pràctiques**

- S**istemes de resposta a preguntes:** Interacció amb assistents intel·ligents per obtenir informació concreta.
- **Traducció automàtica:** Converteix textos d'un idioma a un altre mantenint significat i naturalitat.
- **Anàlisi de sentiments:** Detecta opinions i emocions en textos, útil en xarxes socials i atenció al client.

- **Extracció de coneixement:** Transformació de textos no estructurats en dades per a grafs i bases de coneixement.
- **Resum automàtic**: Genera versions breus i rellevants de documents llargs.

Tecnologies i eines

El PLN actual es recolza en **aprendre automàtic profund (*Deep Learning*)** i **models de transformadors**, que permeten representacions complexes del llenguatge.

Eines i frameworks habituals:

- **spaCy**: processament i etiquetatge de text.
- **Amazon Comprehend**: identificació d'entitats, relacions i sentiments.
- **Hugging Face Transformers**: models de llenguatge preentrenats i adaptables.

En el projecte: aquestes tecnologies s'utilitzen per extreure **entitats, relacions i atributs** dels documents i preparar el context per al model generatiu (LLM).

4.4 Models generatius i LLMs

Els **models generatius de llenguatge** són sistemes d'intel·ligència artificial dissenyats per generar text coherent i contextualitzat a partir d'una entrada determinada. Els **models de llenguatge grans (LLMs, *Large Language Models*)**, com **GPT (Generative Pre-trained Transformer)**, **Claude** o **PaLM**, són exemples destacats i han revolucionat el camp del **Processament del Llenguatge Natural (PLN).**

Funcionament bàsic

Un model generatiu aprèn una **distribució probabilística sobre seqüències de paraules**. Això li permet predir quina paraula és més probable que segueixi en un context donat i generar text nou coherent i rellevant.

Els models avançats utilitzen arquitectures de **transformadors**, capaces de captar dependències complexes en el text a llarg termini, superant limitacions d'antics models seqüencials com **RNN** o **LSTM [14]**.

Preentrenament i ajust (*fine-tuning*)

Els LLMs és preentrenat amb grans quantitats de dades no etiquetades (internet, llibres, articles), aprenent patrons lingüístics i coneixements generals.

Posteriorment, es poden ajustar per tasques específiques o utilitzar en:

- **Zero-shot:** executar tasques sense exemples.
- **Few-shot:** executar tasques amb pocs exemples.

Capacitats i usos

- **Generació de text:** respostes automàtiques, redacció creativa, articles.
- **Resum de documents:** crear versions breus i rellevants de textos llargs.
- **Traducció automàtica:** mantenint precisió i context.

- **Extracció i anàlisi de dades:** complement a mètodes tradicionals.
- **Assistència conversacional:** chatbots i assistents virtuals capaços de mantenir diàlegs naturals.

Limitacions i reptes

- **Hallucinations:** poden generar informació inexacta o inventada.
- **Recursos computacionals:** entrenament i inferència molt exigents.
- **Dependència de dades:** poden reflectir biaixos i errors dels conjunts d'entrenament.
- **Privacitat i seguretat:** risc de revelar informació delicada o vulnerabilitat a manipulacions.

Aplicació en el projecte En aquest projecte s'utilitza **Claude 3.5 Sonnet**, accessible via **AWS Bedrock** amb el perfil d'empresa **aida-dev**.

El model:

Genera respostes contextualitzades i enriquides.

Actua com a agent verificador i generador, validant, corregint i ampliant el coneixement extret dels documents tècnics.

Això combina la capacitat de **generació de llenguatge natural dels LLMs** amb una base sòlida i actualitzada de coneixement gràcies a la **recuperació prèvia** i el **processament semàntic de dades.**

4.5 Ecosistema tecnològic i eines emprades

Per dur a terme aquest projecte s'ha utilitzat un conjunt de **tecnologies punteres** que faciliten l'extracció, processament i organització d'informació a partir de documents PDF i la interacció amb **models de llenguatge avançats (LLMs)**.

Neo4j

Neo4j és una base de dades orientada a **grafs**, dissenyada per emmagatzemar i consultar dades amb relacions complexes. En lloc de taules, utilitza **nodes**, **arestes** i **propietats** per representar entitats i connexions.

Aplicació al projecte:

Les entitats extretes dels PDFs es representen com a nodes i les relacions entre elles com a arestes. Això permet navegar pel coneixement i fer consultes complexes de manera natural.

Avantatges:

- Representació flexible de dades semàntiques.
- Llenguatge de consulta expressiu: **Cypher**.
- Escalabilitat i eficiència en consultes multicapa.

spaCy

spaCy és una biblioteca de **PLN (Processament del Llenguatge Natural)** en Python, amb eines per a anàlisi sintàctica, extracció d'entitats, etiquetatge gramatical i més.

Aplicació al projecte:

Processa el text dels PDFs, segmenta oracions i paraules, identifica dependències sintàctiques i facilita l'extracció d'entitats i relacions.

Avantatges:

- Processament ràpid i eficient.
- Models preentrenats en diversos idiomes.
- Gran varietat d'eines per a anàlisi profunda del text.

Amazon Comprehend

Amazon Comprehend és un servei gestionat de PLN que utilitza **aprenentatge automàtic** per identificar entitats, llenguatge, sentiments, temes i relacions en text.

Aplicació al projecte:

Extracció automàtica d'entitats, frases clau i sentiments dels textos. Externalitza la complexitat de l'entrenament i manteniment dels models.

Avantatges:

Servei gestionat, escalable i fàcil d'integrar.

Processa textos en diversos idiomes.

Genera metadades que enriqueixen els grafs de coneixement.

Python i llibreries auxiliars

El projecte es basa en **Python**, aprofitant la seva versatilitat i l'ecosistema de llibreries per manipulació de fitxers, processament de text i gestió de dades.

Llibreries utilitzades:

- **json**: serialització i emmagatzematge de resultats.
- **concurrent.futures:** processament paral·lel de pàgines.
- **py2neo:** connexió i manipulació del graf Neo4j.
- **spaCy i altres paquets PLN** per l'extracció semàntica.
- **Boto3:** interacció amb serveis AWS.

5 Requisits funcionals i no funcionals

5.1 Requisits funcionals

Els requisits funcionals descriuen què ha de fer el sistema des del punt de vista de l'usuari. Es presenten en forma de **casos d'ús:**

Cas d'ús 1: Carregar document no estructurat

Actor principal: Tècnic o usuari.

Descripció: El sistema permet carregar un document en format **PDF** (Portable Document Format) per processar-lo automàticament.

Entrada: Arxiu **PDF**.

Sortida: Informació estructurada a nivell de frases i entitats.

Excepcions: Error de lectura o format no vàlid.

Cas d'ús 2: Extracció d'informació estructurada

Actor principal: Sistema.

Descripció: El sistema extreu informació a nivell de frases, entitats, atributs i relacions semàntiques.

Entrada: Text pla del document.

Sortida: Nodes i relacions per al graf.

Cas d'ús 3: Validació automàtica del coneixement

Actor principal: Sistema.

Descripció: Un agent de verificació basat en **LLM** valida que la informació extreta sigui coherent i útil per a la consulta.

Entrada: Resultats de l'extracció.

Sortida: Graf corregit o marcat com incomplet.

Cas d'ús 4: Construcció i població del graf

Actor principal: Sistema.

Descripció: La informació validada s'integra a **Neo4j** seguint una jerarquia: *Document → Sentència → Entitat → Relació.*

Entrada: Informació estructurada.

Sortida: Graf en Neo4j.

Cas d'ús 5: Consulta semàntica al graf

Actor principal: Tècnic o usuari.

Descripció: L'usuari pot formular preguntes en llenguatge natural. Aquestes es tradueixen a cerques al graf amb paraules clau i patrons.

Entrada: Pregunta en llenguatge natural.

Sortida: Resposta textual amb cites de les frases rellevants del graf.

Cas d'ús 6: Generació de resposta augmentada (*RAG*)

Actor principal: Sistema.

Descripció: El sistema genera una resposta a partir de les dades recuperades i les envia a un **LLM** (Claude Sonnet via **Amazon Bedrock**) per elaborar una explicació detallada.

Entrada: Context rellevant + pregunta.

Sortida: Resposta augmentada i contextualitzada.

5.2 Requisits no funcionals

Els requisits no funcionals estableixen restriccions i condicions generals del sistema:

Rendiment

- Ha de processar documents extensos en temps raonable.
- Les consultes al graf han de respondre en menys de 45 segons en condicions normals.

Escalabilitat

L'arquitectura ha de permetre processar documents més grans.

Ha de permetre afegir nous documents al graf de manera incremental.

Compatibilitat

- El sistema utilitza Neo4j com a base de coneixement i LLMs via Amazon Bedrock.
- Compatible amb entorns locals i de núvol (*cloud*).

Traçabilitat i aplicabilitat

- Les respostes han d'incloure referències explícites a les frases del document.
- Ha de garantir justificació semàntica del coneixement usat.

Seguretat i privadesa

Ha de protegir la confidencialitat dels documents carregats, especialment en context industrial (per exemple, IDIADA).

La transmissió de dades a serveis externs com Bedrock ha d'estar xifrada.

Mantenibilitat i extensibilitat

- Arquitectura modular per facilitar l'actualització de components (extractor, agent de validació, models de consulta).
- Ha de ser fàcil afegir noves funcionalitats, com l'extracció d'imatges o taules.

6 Disseny del sistema

Aquest apartat descriu el disseny de l'arquitectura general de l'aplicació, així com les decisions preses en relació amb la persistència de dades i els criteris generals de modularitat i mantenibilitat. L'aplicació es fonamenta en una arquitectura modular, orientada a processos seqüencials i enfocada a la manipulació i gestió del coneixement extret de documents en format PDF.

6.1 Anàlisi de requisits funcionals

El sistema desenvolupat en el marc del Treball de Fi de Grau té com a objectiu processar documents no estructurats (PDF) i transformar-ne el contingut rellevant en un graf de coneixement. Aquest procés es duu a terme de manera seqüencial i integra diferents fases d'extracció, validació, construcció i consulta. El disseny es basa en una arquitectura modular que facilita la flexibilitat, la reutilització de components i l'anàlisi qualitativa del coneixement obtingut.

6.1.1 Seqüència de processament

La seqüència d'execució típica del sistema segueix els passos següents:

1. **Càrrega del document**: L'usuari incorpora un fitxer PDF i s'inicia el procés de tractament.

2. **Separació i extracció de text**: El sistema divideix el document en pàgines i n'extreu el contingut textual [15].

3. **Extracció de coneixement per pàgina**:

- El mòdul d'extracció utilitza s**paCy** i **Amazon Comprehend** per identificar entitats, relacions i atributs.
- Els resultats s'emmagatzemen en fitxers JSON, un per cada pàgina del document.

4. **Verificació amb LLM:** L'agent verificador basat en un model de llenguatge processa els fitxers JSON i avalua la qualitat del coneixement extret, corregint i ajustant els elements quan cal.

5. **Construcció del graf**: El mòdul corresponent transforma la informació verificada en nodes i relacions, i la carrega a la base de dades Neo4j.

6. **Consulta i generació de respostes:** Quan l'usuari formula una pregunta, el mòdul de consulta contextualitza la demanda i genera una resposta fonamentada en el coneixement representat al graf.

6.2 Arquitectura de l'aplicació

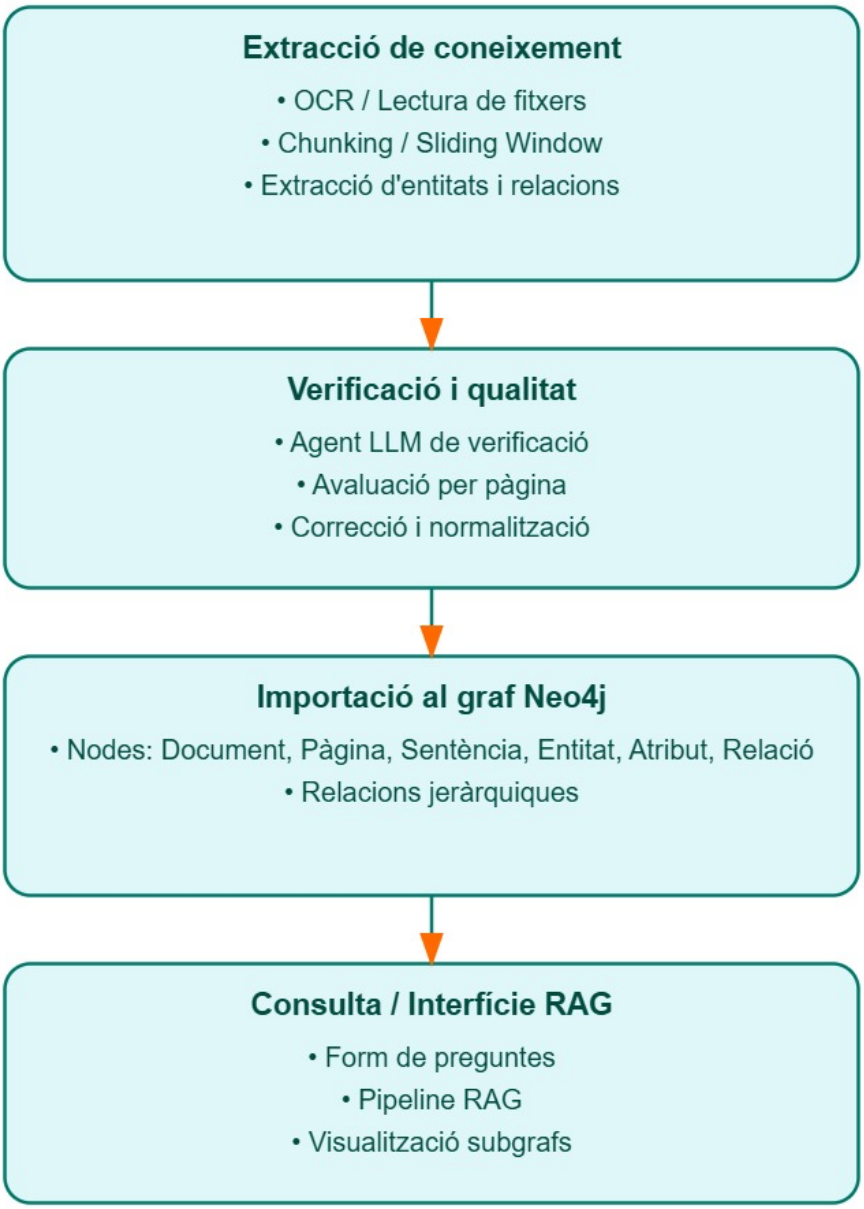


Figura 1. Representació del flux del sistema

6.2.1 Inicialització i integració de components

Aquest subapartat descriu el procés d'inicialització dels diferents mòduls que formen part de l'arquitectura i la seva integració amb serveis externs. L'objectiu és assegurar que el sistema disposi de tots els recursos necessaris per dur a terme el cicle complet de processament: des de l'extracció de contingut fins a la verificació i enriquiment del coneixement.

1. **Extracció de contingut**

El sistema importa diferents extractors del mòdul `file_extractor.extractors`, entre els quals:

AgentOCRExtractor: component basat en OCR intel·ligent que actua com a reforç en casos de baixa qualitat d'imatge.

Aquesta varietat d'extractors permet aplicar estratègies híbrides per maximitzar la qualitat del text extret.

2. **Emmagatzematge i persistència**

El sistema suporta dos mecanismes d'emmagatzematge:

- *LocalStorage*: destinat a proves locals o entorns de desenvolupament.

3. **Processament de PDF i paral·lelització**

Mitjançant la classe `PdfProcessor` i la llibreria `concurrent.futures.ThreadPoolExecutor`, el sistema és capaç de processar múltiples pàgines en paral·lel, optimitzant els temps de resposta i millorant l'escalabilitat en documents extensos.

4. **Integració amb AWS i serveis de llenguatge**

- *Bedrock*: es crea un client per al servei `bedrock-runtime` d'AWS (perfil **aida-dev**), que permet la connexió amb models generatius (LLMs) per a tasques de verificació i enriquiment.
- *Comprehend*: s'inicialitza un client d'AWS Comprehend per a l'extracció automàtica d'entitats i relacions.
- *spaCy*: s'incorpora el model `es_core_news_md` per al processament de text en castellà (o `en_core_web_md` per documents en anglès), amb l'objectiu de complementar la detecció d'entitats i relacions.

5. **Eines de suport i definició d'agents**

Finalment, es carreguen eines com `FormatterTool` i esquemes de configuració (`AWSLLMConfig`, `LLMParameters`, `ToolConfig`), que defineixen la interacció dels agents intel·ligents amb els models de llenguatge i permeten adaptar els paràmetres d'execució a cada necessitat.

En conjunt, aquest bloc d'inicialització garanteix la disponibilitat de recursos mínims per processar documents, analitzar contingut textual, extreure coneixement i delegar en un agent LLM les tasques de verificació i enriquiment. A més, aquesta capa d'integració ofereix un alt grau de modularitat, afavorint el manteniment i facilitant la incorporació de nous serveis en el futur.

6.2.2 Procés de processament de documents i extracció de coneixement

Aquest apartat descriu el funcionament intern del mòdul encarregat de processar els documents i obtenir-ne informació rellevant (entitats, relacions, atributs, sentiments i frases clau). El codi presentat implementa un pipeline complet, en què intervenen serveis externs (AWS Comprehend), llibreries locals (spaCy) i mecanismes d'execució paral·lela per escalar el rendiment.

1. Configuració de l'emmagatzematge i processament bàsic

El sistema defineix un gestor d'emmagatzematge local amb LocalStorage, on es desaran els resultats i fitxers intermedis. A continuació, es crea un objecte PdfProcessor que:

• Estableix paràmetres com la resolució (150 dpi) i desactiva l'extracció d'imatges per centrar-se en el text.

• Utilitza AgentOCRExtractor amb un model LLM d'AWS Bedrock (Anthropic Sonnet 3.7) per enriquir l'extracció textual en casos on l'OCR és necessari.

• Permet l'execució en paral·lel (execution_mode="threaded", max_workers=16) per processar múltiples pàgines de manera eficient.

2. Extracció de contingut amb AWS Comprehend

Un cop processat el text, es defineixen diferents funcions per extreure coneixement:

• extract_entities: identifica entitats nominals (persones, organitzacions, llocs, etc.) utilitzant Comprehend. Es filtren només les que superen un llindar de confiança (>0.1).

• extract_key_phrases: detecta frases clau amb Comprehend i en conserva les més rellevants (confiança >0.7) [16].

3. Anàlisi sintàctica i detecció de relacions amb spaCy

Per complementar Comprehend, s'utilitza spaCy amb el model en castellà o anglés, si es desitja un altre idioma es canvia el paràmetre i es descarregar el model de Spacy amb el idioma que vulguis adaptar.

• Es defineix una funció extract_relations_between_entities que analitza la sintaxi de les frases per detectar relacions entre entitats basant-se en les estructures subjecte – verb – objecte.

• També es contemplen clàusules relatives i estructures compostes, assegurant una detecció robusta de vincles semàntics.

• Les relacions duplicades s'eliminen per mantenir només vincles únics i consistents.

4. Extracció d'atributs

Mitjançant la funció extract_attributes, s'afegeixen propietats addicionals a les entitats (adjectius, quantitats, qualificatius). Això permet enriquir el coneixement amb descripcions més fines, útils per construir un graf semàntic més complet.

5. Fusió d'entitats i frases clau

Amb la funció merge_entities_and_key_phrases, es garanteix que les frases clau detectades es tractin també com a entitats si encara no hi són presents. Això evita perdre informació rellevant.

6. Processament de pàgines i consolidació de resultats

Cada pàgina es processa amb la funció process_page:

• Es llegeix el text (page_agentic.txt).

• Es passa per spaCy una única vegada per reduir el cost computacional.

• Es genera un conjunt enriquit d'entitats, relacions, atributs, frases clau i sentiments.

• Tot el resultat es desa en format JSON dins la carpeta output/, facilitant-ne la posterior integració al graf de coneixement.

7. Execució paral·lela del pipeline

Finalment, amb la funció process_pages_parallel, s'utilitza un pool de threads per processar totes les pàgines d'un document de manera concurrent (max_workers=10). Això permet escalar el sistema a documents extensos reduint significativament el temps de processament.

Estructura típica d'un fitxer JSON de pàgina:


```
{
    "page": 46,
    "entities": [
        {
            "Score": 0.603,
            "Type": "QUANTITY",
            "Text": "Table 11",
            "BeginOffset": 0,
            "EndOffset": 8,
             "sentence": "Table 11: Application of test requirements for type-approval of brake systems for manufacturers..."
        },
        {
            "Score": 0.779,
            "Type": "QUANTITY",
            "Text": "Table",
            "BeginOffset": 338,
            "EndOffset": 343,
            "sentence": "Required | Not required..."
        }
    ],
    "relations": [
        {
            "subject": {
                "text": "emission type-approval | Tests",
                "type": "KEY_PHRASE",
                "score": 0.997
            },
            "predicate": "Tests at",
            "object": {
                "text": "conformity",
                "type": "KEY_PHRASE",
                "score": 0.998
            },
             "sentence": "| Tests at conformity of production | Tests at in-service conformity...",
            "confidence": 0.997
        }
```

```
        ],
        "attributes": {
            "Table 11": [
                {
                    "attribute": "11",
                    "type": "NUM",
                    "confidence": 0.85
                }
            ]
        },
        "sentences": [
            "Table 11: Application of test requirements for type-approval
of brake systems for manufacturers...",
             "| Tests at conformity of production | Tests at in-service
conformity..."
        ],
        "key_phrases": [
            {
                "Score": 0.935,
                "Text": "Table 11",
                "BeginOffset": 0,
                "EndOffset": 8
            }
        ]
    }
```


• **entities**: llista d'entitats detectades, amb tipus, puntuació de confiança, i la frase completa on apareixen.

• **relations**: llista de relacions entre entitats, amb subjecte, predicate i objecte, a més de la frase on apareix i la confiança associada.

• **attributes**: propietats addicionals associades a les entitats.

• **sentences**: text segmentat en frases per facilitar anàlisis posteriors.

• **key_phrases**: frases clau identificades amb puntuació de rellevància.

D'aquesta manera, cada pàgina genera un fitxer autònom que representa tot el coneixement extraït, permetent tant la verificació automàtica com la seva posterior càrrega en el graf de coneixement.

6.2.3 Avaluació de qualitat del coneixement extret

El mòdul de verificació de coneixement té com a objectiu principal avaluar i validar la qualitat de les entitats, atributs i relacions que han estat extretes prèviament d'un document [18]. Aquesta fase actua com un filtre de control de qualitat, assegurant que la informació incorporada al graf de coneixement sigui coherent, fiable i completa.

Entrada del mòdul

El mòdul rep com a entrada els resultats d'extracció generats anteriorment. Els resultats proporcionats inclouen:

• Les entitats detectades, amb el seu tipus i text associat.

• Els atributs vinculats a les entitats, amb el nom i el valor corresponent.

• Les relacions entre entitats, juntament amb el predicat que les connecta.

• Els fragments de text de context on apareixen aquests elements.

• Una puntuació de confiança inicial assignada durant la fase d'extracció.

De manera opcional, també es pot utilitzar el document original per recuperar frases de context en cas que no estiguin disponibles als resultats.

Procés de validació

Per analitzar aquesta informació, el mòdul utilitza models Pydantic que representen formalment els diferents tipus d'elements (entitats, atributs i relacions) [17]. Gràcies a aquests models, és possible:

• Validar automàticament la coherència i integritat de les dades.

• Comprovar que tots els camps obligatoris estiguin presents.

• Avaluar cada element com a correcte o incorrecte, basant-se en regles definides al system prompt del agent [19].

Quan un element es considera incorrecte, el sistema pot generar propostes de correcció, com ara:

• un tipus d'entitat més apropiat,

• un valor d'atribut més ajustat,

• o un predicat alternatiu per a una relació.

A més, cada avaluació queda justificada amb la frase de context corresponent i amb una explicació breu del motiu de la decisió.

Organització i resultats

Un cop finalitzada la validació, el mòdul:

• Classifica els elements en llistes d'entitats, atributs i relacions correctes.

• Incorpora aquelles entitats que apareixen només dins d'atributs, per evitar la pèrdua d'informació.

• Elimina frases de context duplicades per reduir redundàncies.

A continuació, calcula un conjunt d'estadístiques generals [20], que inclouen:

• el nombre total d'elements processats,

• el nombre i percentatge d'elements correctes,

• i la precisió global del coneixement validat.

Tota aquesta informació s'integra dins d'un model de resultat final, que encapsula tant les dades validades com les mètriques de qualitat.

Sortida del mòdul

El resultat de la validació es materialitza en un fitxer JSON, que conté:

• Les llistes d'entitats, atributs i relacions amb els camps de correcció.

• Les frases de context i explicacions associades.

• Les estadístiques generals i una valoració global del coneixement.

Aquest fitxer es desa automàticament en una carpeta de sortida i per posteriorment ser utilitzat per a la construcció del graf de coneixement,

En resum, aquest mòdul actua com un component central de validació i control de qualitat dins del sistema, assegurant que el coneixement extret sigui coherent, verificat i apte per a la seva incorporació a les fases posteriors.

Estructura típica del JSON de verificació:


```
{
  "text_sample": "Official Journal of the European Union ...",
  "entity_count": 15,
  "correct_entities": 13,
  "attribute_count": 10,
  "correct_attributes": 10,
  "relation_count": 5,
  "correct_relations": 5,
  "overall_accuracy": 74.29,
  "entity_evaluations": [
    {
      "entity_text": "European Union",
      "entity_type": "ORGANIZATION",
      "score": 0.9327,
      "is_correct": true,
      "suggested_type": null,
      "explanation": "VALIDACIÓN: La clasificación como ORGANIZATION
es correcta.",
      "sentence": "of the European Union"
    }
  ],
  "attribute_evaluations": [
    {
      "entity": "REGULATION (EU) 2024/1257",
      "attribute": "type",
      "type": "classification",
      "confidence": 0.95,
      "is_correct": true,
      "suggested_attribute": null,
      "explanation": "Se añade el atributo 'type' con valor
'regulation'.",
      "sentence": null
    }
  ],
  "relation_evaluations": [
    {
```

```
            "subject": "REGULATION (EU) 2024/1257",
            "predicate": "amends",
            "object": "Regulation (EU) 2018/858",
            "confidence": 0.95,
            "is_correct": true,
            "suggested_predicate": null,
            "explanation": "Se crea esta relación para reflejar que la nueva
regulación modifica la anterior.",
            "sentence": null
          }
        ],
        "overall_assessment": "Evaluación completada con batches",
        "sentences": [
          "of the European Union",
          "amending Regulation (EU) 2018/858 ..."
        ]
      }
```


• **entity_count / correct_entities**: nombre total d'entitats detectades i nombre correctes segons la verificació.

• **attribute_count / correct_attributes**: atributs detectats i correctes.

• **relation_count / correct_relations:** relacions detectades i correctes.

• **overall_accuracy**: percentatge global de precisió del coneixement extret.

• **entity_evaluations**: informació detallada per cada entitat, incloent si és correcta, suggeriments i frase on apareix.

• **attribute_evaluations**: atributs verificats amb confiança i possibles correccions.

• **relation_evaluations**: relacions verificades amb indicació de subjecte, objecte, predicate, confiança i explicació.

• **overall_assessment:** resum del procés de verificació.

• **sentences**: text segmentat utilitzat per la verificació.

Aquest mecanisme permet tenir una avaluació quantitativa i qualitativa del coneixement extret, facilitant la detecció d'errors i la millora del model d'extracció.

6.2.4 Importació i modelatge del coneixement verificat al graf Neo4j

Aquesta part del projecte s'encarrega d'importar al graf Neo4j tot el coneixement verificat que s'ha extret dels documents, incloent-hi entitats, atributs, relacions, frases i referències a taules [21][22]. L'objectiu principal és transformar els resultats de la verificació (fitxers JSON generats a l'etapa anterior) en nodes i relacions dins del graf, de manera que es pugui **navegar i consultar el coneixement estructurat [23]**.

La connexió amb Neo4j es realitza mitjançant la biblioteca py2neo, establint un accés directe al servidor amb les credencials corresponents [24].

Creació del node Document

El procés s'inicia amb la creació d'un node principal de tipus **Document**, que representa el document complet que s'està processant. Aquest node actua com a arrel per a totes les pàgines i elements que s'importaran posteriorment.

Importació de pàgines

El codi cerca tots els fitxers JSON de verificació de cada pàgina en un directori específic, els ordena i en recorre el contingut per tal d'importar-lo [25]. La funció principal import_verified_entities_and_relations s'encarrega de la importació. Primer comprova que cada fitxer existeixi i es pugui llegir correctament com a JSON. A continuació, crea un node de tipus **Page** per representar la pàgina concreta, assignant-hi les estadístiques de qualitat del coneixement:

• Nombre total d'entitats, atributs i relacions.

• Nombre d'elements correctes.

• Precisió global.

Cada node Page es connecta al node Document mitjançant la relació HAS_PAGE.

Gestió de frases (Sentence)

Un dels punts clau és la gestió de les frases.

Es manté un diccionari intern per garantir que cada frase es crea només una vegada.

• Quan una entitat o relació fa referència a una frase concreta, es crea un node Sentence.

• Aquest node es vincula amb el node Page i amb les entitats que conté, mitjançant la relació CONTAINS_ENTITY.

D'aquesta manera, es preserva el context textual de cada element dins del graf [26].

Creació d'entitats

Per cada entitat verificada, es genera un node Entity que inclou:

• Propietats originals.

• Propietats de verificació: tipus suggerit, explicació, puntuació de confiança i estat de correcció.

• Atributs associats (valor i tipus).

Quan una entitat té una frase associada, es vincula amb el node Sentence corresponent [27][28].

Creació de relacions

Les relacions connecten dues entitats mitjançant un predicat normalitzat, amb mesures de protecció contra:

• Caràcters no vàlids.

• Noms massa llargs.

• Predicats que comencin amb números.

Cada relació guarda informació addicional: puntuació de confiança, explicació i estat de correcció. Quan una relació té associada una frase, aquesta es vincula amb ambdues entitats per tal de preservar-ne el context.

Resultat final

Al final del procés, cada pàgina del document queda representada al graf amb els seus:

• Nodes de Sentence, Entity, Relation, Attribute i Table.

• Connexions correctes amb el node Document arrel. Això permet:

• Realitzar consultes semàntiques.

• Explorar la informació de forma jeràrquica i estructurada.

• Fer anàlisi posterior de la qualitat i integritat del coneixement importat.

Aquesta etapa integra completament la verificació dins de l'arquitectura general de l'aplicació, proporcionant un graf de coneixement consistent i consultable.

6.2.5 Pipeline de consulta i resposta sobre el graf de coneixement amb LLM

Aquest mòdul permet la interacció directa amb el graf de coneixement emmagatzemat a Neo4j i l'obtenció de respostes contextualitzades mitjançant un model de llenguatge desplegat a través d'Amazon Bedrock. L'objectiu és assegurar que les respostes siguin sempre traçables a la informació continguda al graf i no depenguin únicament de la generació del model.

El flux de treball consta de tres etapes principals:

1. **Anàlisi de la consulta de l'usuari**
    - Normalització del text (neteja d'accents, puntuació i paraules buides).
    - Lematització amb *spaCy*.
    - Extracció de paraules clau en mode estricte o laxo mitjançant `analyze_query`.
2. **Recuperació de context des del graf Neo4j**
    - Construcció de consultes *Cypher* amb `retrieve_context` utilitzant les keywords [29].
    - Filtratge per secció numèrica si la consulta ho especifica.
    - Conversió del resultat a format Markdown amb `context_to_markdown` per preparar-lo com a entrada al model.
3. **Generació de la resposta amb Bedrock**
    - Creació d'un *prompt* estructurat amb missatge de sistema i d'usuari.
    - Crida al model **Claude Sonnet** via Bedrock mitjançant `call_bedrock`.
    - Resposta final amb explicacions i cites de sentències o indicació de manca d'informació [30].

La funció principal `answer_query_with_graph` orquestra tot el procés: primer prova l'anàlisi estricta, i en cas de no obtenir resultats, aplica el mode laxo. Finalment, retorna la

resposta generada alhora que permet interacció iterativa amb l'usuari en un bucle principal. Aquest pipeline combina tècniques de **PLN, recuperació semàntica i generació condicionada** i constitueix la peça clau per a l'explotació pràctica del coneixement representat al graf.

6.2.6 Processament, verificació i anàlisi de preguntes sobre el graf de coneixement

Aquest conjunt de funcions té com a objectiu avaluar l'eficàcia del sistema de consulta mitjançant un procés automatitzat de preguntes-respostes i verificació. El flux segueix quatre fases principals:

1. **Lectura i processament de preguntes**
    - `read_questions_from_csv`: llegeix preguntes des de fitxers CSV.
    - `process_questions_from_csv`: envia cada pregunta al mòdul `answer_query_with_graph` i desa resultats en JSON.
2. **Execució global sobre múltiples fitxers**
    - `process_all_questions_and_save_results`: processa tots els CSV numerats d'un directori i desa respostes per pàgina.
    - Registre de logs detallats per traçabilitat.
3. **Verificació de respostes amb LLM**
    - `create_answer_verification_agent`: crea un agent que compara la resposta amb el text original de la pàgina.
    - `verify_results_for_all_pages`: genera per a cada resposta un JSON amb camps `is_correct`, `justification` i `evidence`.
4. **Anàlisi estadística de l'efectivitat**
    - `study_query_effectiveness`: calcula la precisió (correctes/incorrectes/errors), tant per pàgina com globalment.

Aquest procés proporciona una **avaluació quantitativa i qualitativa** de la capacitat del sistema per generar respostes correctes a partir del graf, i permet identificar punts de millora en l'extracció o la recuperació de coneixement [31].

6.3 Model de llenguatge

Un aspecte clau d'aquesta arquitectura és la tria del model de llenguatge dins del mòdul de validació i de consulta. Concretament, s'ha optat per **anthropic.claude-3-5-sonnet-20240620-v1:0** a través de Bedrock (AWS) [32]. La decisió es fonamenta en diversos motius:

• **Precisió i capacitat de raonament**: aquest model és especialment robust en tasques de verificació i anàlisi crític, aspecte essencial per a la validació d'entitats i relacions sense introduir informació nova no present al text original.

• **Equilibri entre cost i rendiment [33]:** ofereix un punt intermedi òptim entre models més lleugers (més econòmics però menys precisos) i models més grans (molt més costosos computacionalment).

• **Integració nativa amb AWS Bedrock [34]:** simplifica la gestió d'infraestructura, permetent escalar fàcilment el sistema i mantenint una arquitectura homogènia amb la resta de serveis AWS utilitzats (Comprehend, S3, etc.).

• **Limitacions:** el model no sempre és consistent en tasques molt específiques de domini tècnic i requereix un disseny acurat dels prompts per garantir la qualitat de les respostes. Tot i això, l'ús de Pydantic i esquemes de validació ha permès minimitzar aquests riscos [35].

Aquesta elecció ha estat, per tant, estratègica per garantir fiabilitat, escalabilitat i coherència en el pipeline de processament.

6.4 Disseny de la persistència de dades (BD)

El sistema fa ús de diversos mecanismes de persistència, cadascun adequat al seu propòsit:

1. Fitxers JSON locals

• Utilitzats per guardar les sortides d'extracció i validació de coneixement per pàgina [36].

• Són fàcils d'inspeccionar, modificar i traslladar, cosa que afavoreix el desenvolupament i el testing.

2. Neo4j com a base de dades de coneixement [37]

• Representa la informació validada en forma de graf.

• Utilitza etiquetes de nodes (Document, Sentence, Entity, Attribute, Relation) i relacions (CONTAINS_SENTENCE, CONTAINS_ENTITY, HAS_ATTRIBUTE, SUBJECT_OF, OBJECT_IS…).

• Permet consultes complexes mitjançant Cypher i visualització gràfica intuïtiva [38].

Aquesta combinació permet flexibilitat en l'anàlisi i traçabilitat del coneixement generat, alhora que facilita l'evolució del projecte cap a entorns més robustos (com una API o una BD documental en cas necessari).

7 Implementació

Aquest apartat descriu com s'ha implementat tècnicament el sistema, incloent les tecnologies emprades, els algorismes específics utilitzats en cada fase, i l'organització general del codi. El projecte es basa en una arquitectura modular en Python i fa ús d'eines de processament de llenguatge natural, gràfics de coneixement i serveis de núvol.

7.1 Tecnologies utilitzades

L'implementació ha estat desenvolupada íntegrament en Python 3.10, amb suport de diverses biblioteques i serveis:

• **Amazon Comprehend (AWS):** per a l'extracció inicial d'entitats i relacions [39].

• **spaCy**: per a la detecció complementària d'entitats i frases clau [40].

• **Py2neo**: per a la comunicació amb Neo4j i construcció del graf.

• **Neo4j**: base de dades de graf per representar el coneixement estructurat.

• **LLM (via Bedrock amb Claude):** per validar i enriquir les entitats i relacions detectades.

• **Pydantic**: per validar estructures de dades i generar els resultats de qualitat de manera consistent.

• **JSON i OS:** per a la gestió de fitxers temporals i intermedis.

Per a una descripció més detallada i específica de cada tecnologia i la seva funció dins del projecte, es pot consultar l'apartat **4.5 Ecosistema tecnològic i eines emprades** de l'apartat Fonaments Teòrics.

Tot el codi es troba organitzat en mòduls independents, cada un corresponent a una etapa del pipeline: extracció, validació, construcció del graf, i consulta/resposta.

7.2 Algorismes específics

A continuació es detallen alguns dels algoritmes i estratègies més destacades:

a) Extracció per pàgines

Per a cada PDF, el sistema processa cada pàgina de manera independent, extraient el text, passant-lo per Comprehend i spaCy, i generant fitxers .json amb les entitats, relacions i atributs trobats. Aquesta granularitat per pàgina millora la traçabilitat i la precisió durant la validació.

b) Validació de coneixement (Agent LLM) [41]

Es fa ús d'un agent LLM per revisar el coneixement extret, segons el següent procés:

• Se li passa el text original de la pàgina + el JSON amb el coneixement detectat.

• L'agent analitza si les entitats, atributs i relacions són correctes i les marca com a correctes, incorrectes o dubtoses, donant justificacions.

• A més, retorna metadades com el nombre d'errors, precisió, i una valoració qualitativa de la qualitat del coneixement extret.

Aquest procés permet garantir que només s'insereixi coneixement verificat al graf.

c) Construcció del graf jeràrquic [42]

S'implementa una jerarquia clara:

- Document
  └── Sentence
    └── Entity
      └── Relation

Cada node conté identificadors únics, referències al document i a la pàgina, i estat de verificació. Les relacions semàntiques (HAS_ATTRIBUTE, SUBJECT_OF, etc.) s'afegeixen només si passen la validació.

d) Consulta i resposta amb recuperació [43]

Per respondre preguntes:

1. Es fa una anàlisi semàntica de la pregunta per identificar temes o entitats claus.

2. Es recupera context rellevant del graf (en lloc d'un context textual).

3. Es genera una resposta basada en aquest coneixement estructurat, que sovint és més precís que un context no estructurat.

4. Un agent de verificació pot avaluar si la resposta és coherent amb el graf. Aquest enfoc garanteix la transparència, traçabilitat i precisió de les respostes.

7.3 Estratègies d'escalabilitat i mantenibilitat

• L'ús de fitxers JSON intermedis permet repetir o depurar qualsevol fase sense executar tot el pipeline.

• L'agent de verificació és modular i pot ser substituït o combinat amb altres models.

• El disseny del graf permet fàcilment afegir nous tipus de nodes (per exemple, per a imatges i taules).

• El sistema està preparat per a execució per lots i, si cal, per a paral·lelització.

8 Avaluació i proves

Aquest apartat presenta el disseny i els resultats de les proves realitzades per validar el funcionament del sistema desenvolupat. Les proves cobreixen tant la part funcional com la qualitat dels resultats obtinguts al llarg de cada etapa del pipeline: extracció, verificació, construcció del graf i recuperació de coneixement per a preguntes.

8.1 Estratègia general d'avaluació

L'estratègia d'avaluació del sistema s'ha plantejat de manera integral, combinant diverses perspectives per assegurar que el sistema no només funciona correctament, sinó que també proporciona resultats útils i fiables [44]. Concretament, l'avaluació es centra en dues grans dimensions:

1. Validació funcional del sistema:

Aquesta part de l'avaluació té com a objectiu verificar que cada component implementat compleix amb la funcionalitat especificada en els requisits. Es comprova, per exemple, que:

• L'extracció de text i el processament dels documents PDF s'executa sense errors i que el text obtingut és complet i coherent.

• L'extracció d'entitats, atributs i relacions genera les estructures esperades i les guarda correctament en el graf Neo4j.

• El pipeline de consultes sobre el graf retorna context rellevant i coherent respecte a la pregunta introduïda.

• La interacció amb l'agent LLM produeix respostes que poden ser interpretades i utilitzades per a la verificació i avaluació [45].

Aquesta fase assegura que tots els mòduls funcionen individualment i de manera integrada dins del sistema global, prevenint errors d'execució i inconsistències entre components.

2. Avaluació de qualitat del coneixement extret:

Una vegada verificat que el sistema funciona correctament, es realitza una avaluació més qualitativa sobre el coneixement que el sistema ha extret i modelat. Aquesta avaluació es basa en l'ús d'un agent **LLM** dissenyat específicament per:

• Comparar les respostes generades a partir del graf amb el contingut original de cada pàgina de text.

• Determinar si les respostes són correctes, incompletes o incorrectes, oferint una justificació i citant fragments textuals que serveixen com a evidència.

• Mesurar la precisió, és a dir, el percentatge de respostes correctes respecte al total de preguntes processades.

• Avaluar la cobertura, comprovar que el sistema és capaç de respondre a la majoria de preguntes rellevants formulades.

• Revisar la fiabilitat, observant si els resultats són consistents entre preguntes similars i si la verificació automàtica coincideix amb la informació del text original.

Aquesta doble aproximació permet garantir que el sistema no només és funcional, sinó que també aporta coneixement fiable i útil al usuari final. A més, proporciona una base quantitativa i qualitativa per mesurar el rendiment global del sistema i identificar possibles millores o ajustos en els components.

8.2 Casos de prova dissenyats

Per avaluar el sistema s'han dissenyat diversos casos de prova que permeten comprovar tant el correcte funcionament tècnic com la qualitat del coneixement extret. L'objectiu és provar diferents escenaris en funció de la mida dels documents, l'idioma i la complexitat del contingut [46]. Concretament, s'han utilitzat els següents documents de prova:

1. PDF curt – Curriculum vitae fictici (2 pàgines, castellà):

Aquest document és un curriculum completament inventat d'una persona fictícia, redactat en castellà, amb informació típica com dades personals, experiència laboral, formació i habilitats. L'ús d'aquest PDF permet: • Comprovar que el sistema és capaç d'extreure text i estructurar-lo correctament en pàgines i sentències.

• Validar que les entitats, atributs i relacions identificades (per exemple, noms, dates, llocs de treball) s'emmagatzemen correctament al graf Neo4j.

• Testar la capacitat del pipeline de resposta a consultes sobre un document petit i relativament senzill, assegurant que el sistema funciona amb documents amb poques pàgines i contingut lineal.

2. PDF llarg – Regulació (49 pàgines, anglès, EU 2024/1257):

Aquest document conté la regulació europea fictícia EU 2024/1257, redactada en anglès i amb un volum considerable de contingut tècnic i legal. L'ús d'aquest PDF permet:

• Avaluar el rendiment del sistema en documents extensos, amb múltiples pàgines i estructures complexes (articles, annexos, taules).

• Comprovar que l'extracció automàtica de sentències, entitats i relacions funciona correctament a gran escala i que la informació es modela de manera coherent dins del graf.

• Testar la gestió d'idiomes diferents (anglès vs castellà), garantint que el pipeline de normalització, anàlisi de preguntes i consulta al graf pot operar amb textos en diversos idiomes.

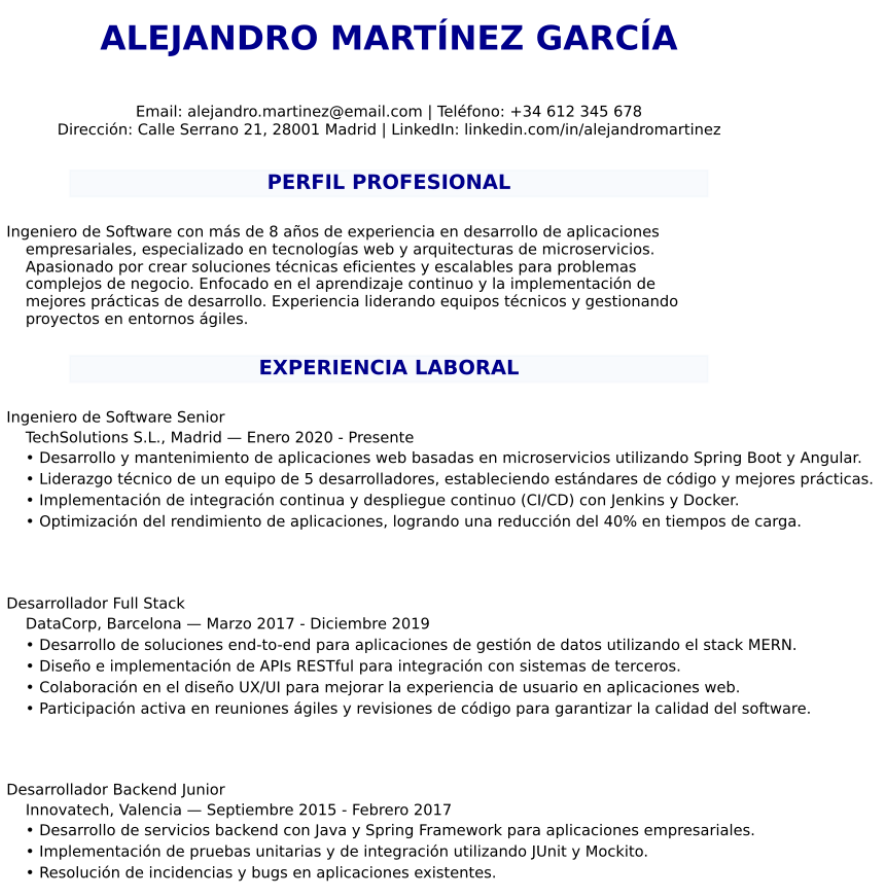
• Analitzar la precisió, cobertura i fiabilitat del sistema amb un document més complex i realista, que simula l'ús del sistema en situacions pràctiques on els documents poden ser extensos i densos.

Aquests dos casos de prova proporcionen un marc de validació complementari: el primer document comprova la robustesa i la funcionalitat amb PDFs petits i simples, mentre que el segon prova l'escalabilitat, la gestió de documents llargs i la compatibilitat amb

diferents idiomes. Aquesta combinació assegura una cobertura ampla en l'avaluació del sistema i evidencia que el projecte es pot utilitzar tant per PDFs més petits com per documents llargs, en anglès o en espanyol, simplement canviant els paràmetres del llenguatge del sistema.

8.2.1 Preparació de preguntes i generació de respostes

Per a facilitar l'avaluació i la verificació del sistema, es crea una carpeta anomenada questions on s'emmagatzema un arxiu CSV per a cada pàgina del PDF testat. Cada CSV conté totes les preguntes que es poden respondre exclusivament amb el contingut d'aquella pàgina concreta. Aquesta estructuració per pàgines permet una comprovació detallada de la capacitat del sistema per recuperar informació específica i limitar l'abast de cada consulta.



ALEJANDRO MARTÍNEZ GARCÍA

Email: alejandro.martinez@email.com | Teléfono: +34 612 345 678
Dirección: Calle Serrano 21, 28001 Madrid | LinkedIn: linkedin.com/in/alejandromartinez

PERFIL PROFESIONAL

Ingeniero de Software con más de 8 años de experiencia en desarrollo de aplicaciones empresariales, especializado en tecnologías web y arquitecturas de microservicios. Apasionado por crear soluciones técnicas eficientes y escalables para problemas complejos de negocio. Enfocado en el aprendizaje continuo y la implementación de mejores prácticas de desarrollo. Experiencia liderando equipos técnicos y gestionando proyectos en entornos ágiles.

EXPERIENCIA LABORAL

Ingeniero de Software Senior
TechSolutions S.L., Madrid — Enero 2020 - Presente
- Desarrollo y mantenimiento de aplicaciones web basadas en microservicios utilizando Spring Boot y Angular.
- Liderazgo técnico de un equipo de 5 desarrolladores, estableciendo estándares de código y mejores prácticas.
- Implementación de integración continua y despliegue continuo (CI/CD) con Jenkins y Docker.
- Optimización del rendimiento de aplicaciones, logrando una reducción del 40% en tiempos de carga.

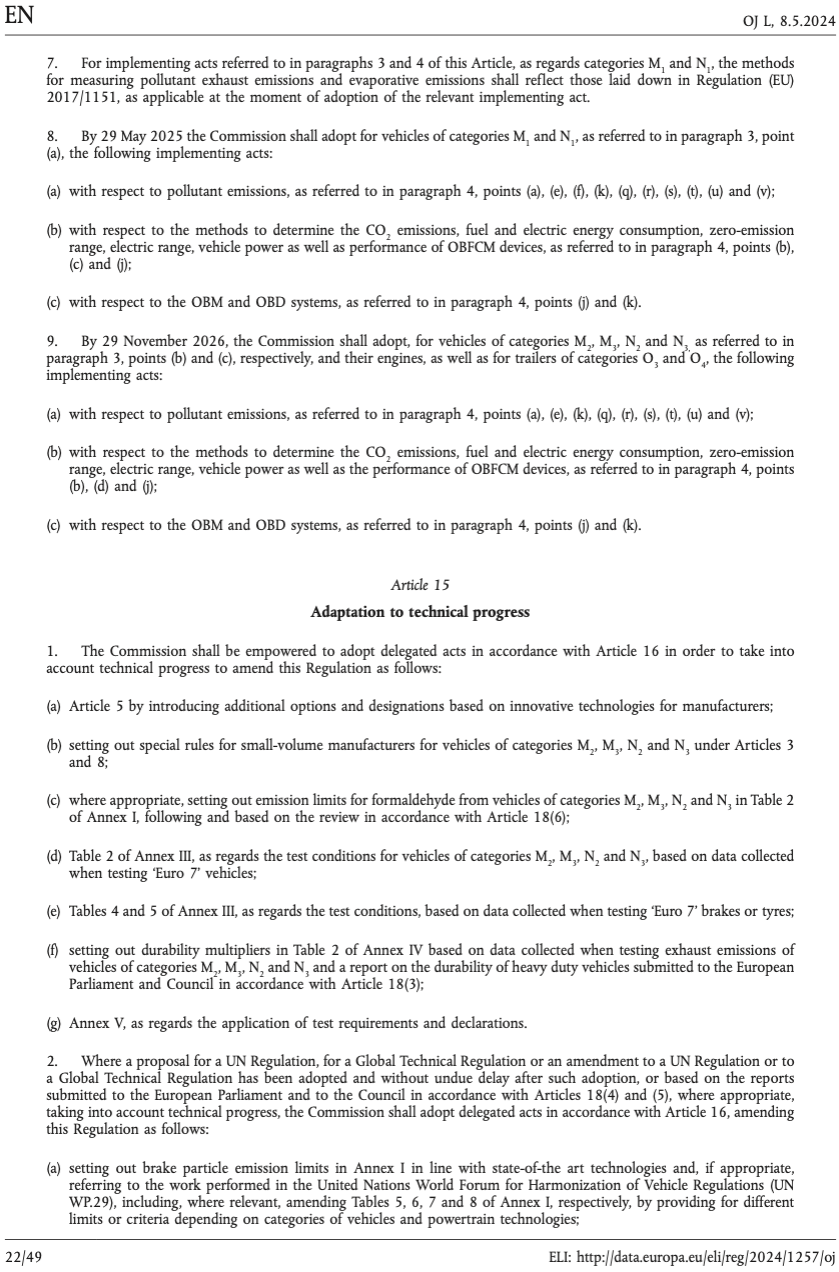
Desarrollador Full Stack
DataCorp, Barcelona — Marzo 2017 - Diciembre 2019
- Desarrollo de soluciones end-to-end para aplicaciones de gestión de datos utilizando el stack MERN.
- Diseño e implementación de APIs RESTful para integración con sistemas de terceros.
- Colaboración en el diseño UX/UI para mejorar la experiencia de usuario en aplicaciones web.
- Participación activa en reuniones ágiles y revisiones de código para garantizar la calidad del software.

Desarrollador Backend Junior
Innovatech, Valencia — Septiembre 2015 - Febrero 2017
- Desarrollo de servicios backend con Java y Spring Framework para aplicaciones empresariales.
- Implementación de pruebas unitarias y de integración utilizando JUnit y Mockito.
- Resolución de incidencias y bugs en aplicaciones existentes.

Figura 2. Primera pàgina del currículum

En el cas del PDF de 49 pàgines sobre la regulació (EU) 2024/1257, s'han escollit de manera aleatòria 5 pàgines per realitzar les proves. D'aquesta manera es poden validar diferents punts del document llarg sense necessitat de processar totes les pàgines, garantint un conjunt de proves representatiu de la seva complexitat i contingut. A més d'aquestes 5

pàgines seleccionades, s'han realitzat proves addicionals amb la consulta interactiva en qualsevol part del document, assegurant que el sistema és capaç de recuperar i verificar informació de manera flexible i sobre qualsevol secció del text.



EN OJ L, 8.5.2024

7. For implementing acts referred to in paragraphs 3 and 4 of this Article, as regards categories M_1 and N_1, the methods for measuring pollutant exhaust emissions and evaporative emissions shall reflect those laid down in Regulation (EU) 2017/1151, as applicable at the moment of adoption of the relevant implementing act.

8. By 29 May 2025 the Commission shall adopt for vehicles of categories M_1 and N_1, as referred to in paragraph 3, point (a), the following implementing acts:

(a) with respect to pollutant emissions, as referred to in paragraph 4, points (a), (e), (f), (k), (q), (r), (s), (t), (u) and (v);

(b) with respect to the methods to determine the CO_2 emissions, fuel and electric energy consumption, zero-emission range, electric range, vehicle power as well as performance of OBFCM devices, as referred to in paragraph 4, points (b), (c) and (j);

(c) with respect to the OBM and OBD systems, as referred to in paragraph 4, points (j) and (k).

9. By 29 November 2026, the Commission shall adopt, for vehicles of categories M_2, M_3, N_2 and N_3, as referred to in paragraph 3, points (b) and (c), respectively, and their engines, as well as for trailers of categories O_3 and O_4, the following implementing acts:

(a) with respect to pollutant emissions, as referred to in paragraph 4, points (a), (e), (k), (q), (r), (s), (t), (u) and (v);

(b) with respect to the methods to determine the CO_2 emissions, fuel and electric energy consumption, zero-emission range, electric range, vehicle power as well as the performance of OBFCM devices, as referred to in paragraph 4, points (b), (d) and (j);

(c) with respect to the OBM and OBD systems, as referred to in paragraph 4, points (j) and (k).

Article 15

Adaptation to technical progress

1. The Commission shall be empowered to adopt delegated acts in accordance with Article 16 in order to take into account technical progress to amend this Regulation as follows:

(a) Article 5 by introducing additional options and designations based on innovative technologies for manufacturers;

(b) setting out special rules for small-volume manufacturers for vehicles of categories M_2, M_3, N_2 and N_3 under Articles 3 and 8;

(c) where appropriate, setting out emission limits for formaldehyde from vehicles of categories M_2, M_3, N_2 and N_3 in Table 2 of Annex I, following and based on the review in accordance with Article 18(6);

(d) Table 2 of Annex III, as regards the test conditions for vehicles of categories M_2, M_3, N_2 and N_3, based on data collected when testing 'Euro 7' vehicles;

(e) Tables 4 and 5 of Annex III, as regards the test conditions, based on data collected when testing 'Euro 7' brakes or tyres;

(f) setting out durability multipliers in Table 2 of Annex IV based on data collected when testing exhaust emissions of vehicles of categories M_2, M_3, N_2 and N_3 and a report on the durability of heavy duty vehicles submitted to the European Parliament and Council in accordance with Article 18(3);

(g) Annex V, as regards the application of test requirements and declarations.

2. Where a proposal for a UN Regulation, for a Global Technical Regulation or an amendment to a UN Regulation or to a Global Technical Regulation has been adopted and without undue delay after such adoption, or based on the reports submitted to the European Parliament and to the Council in accordance with Articles 18(4) and (5), where appropriate, taking into account technical progress, the Commission shall adopt delegated acts in accordance with Article 16, amending this Regulation as follows:

(a) setting out brake particle emission limits in Annex I in line with state-of-the art technologies and, if appropriate, referring to the work performed in the United Nations World Forum for Harmonization of Vehicle Regulations (UN WP.29), including, where relevant, amending Tables 5, 6, 7 and 8 of Annex I, respectively, by providing for different limits or criteria depending on categories of vehicles and powertrain technologies;

22/49 ELI: http://data.europa.eu/eli/reg/2024/1257/oj

Figura 3. Pàgina 22 del PDF de 49 pàgines

Un agent LLM dedicat a la verificació de resultats utilitza aquests CSVs per generar les respostes correctes a cada pregunta. Posteriorment, cada resposta creada és avaluada mitjançant el sistema de consulta interactiva prèviament definit, que realitza una cerca al graf de coneixement. Aquesta metodologia assegura que el sistema pot gestionar preguntes concretes per pàgina, tant en documents curts com en documents llargs, i en diferents idiomes (castellà i anglès), simplement ajustant els paràmetres de llenguatge del sistema.

8.3 Resultats obtinguts

INFO | Found credentials in shared credentials file: ~/.aws/credentials

INFO | 🔑 Keywords extraídas (estricto): ['trabajar', 'programador']

⏳ Pensando...

INFO | 📄 Se recuperaron 42 sentencias del grafo (filtro sección: ninguno)

Sí, Alejandro Martínez García puede trabajar como programador. Basándome en la información proporcionada en el contexto, puedo afirmar lo siguiente:

1. Alejandro tiene formación académica en Ingeniería Informática y un Máster en Ingeniería de Software (1, 2, 17, 18, 19, 22), lo que le proporciona una sólida base teórica para trabajar como programador.

2. Tiene más de 8 años de experiencia en desarrollo de aplicaciones empresariales, especializado en tecnologías web y arquitecturas de microservicios (9, 27). Esto demuestra una amplia experiencia práctica en programación...

Els resultats s'han recollit tant qualitativament com quantitativament, per tal de poder avaluar el rendiment global del sistema i la seva aplicabilitat en diferents contextos:

• Precisió mitjana del coneixement extret i verificat: ~87% en les proves realitzades (segons l'agent verificador). Aquesta xifra reflecteix la capacitat del sistema per identificar entitats, atributs i relacions de manera fiable, fins i tot en documents amb estructures complexes.

• Cobertura mitjana (proporció d'entitats detectades respecte al text original): **~75%.** Això indica que el sistema és capaç de recuperar la major part de la informació rellevant, si bé encara hi ha casos en què certes entitats subtils o relacions implícites poden escapar a la detecció.

• Fiabilitat del sistema: en totes les proves realitzades, els fitxers de sortida (JSON) i els gràfics de coneixement es van generar correctament, i el motor de recuperació de context va respondre de manera consistent a preguntes de dificultat senzilla i mitjana.

• Errors detectats: la majoria provenen de confusions semàntiques (per exemple, interpretar un terme amb més d'un significat possible) o de relacions implícites que no apareixen textualment al document i que, per tant, poden ser formulades de manera incompleta o incorrecta.

En l'exemple del currículum de dues pàgines, es va obtenir un graf relativament compacte però molt dens, amb més de **170** nodes i més de **280** relacions. Aquest graf va permetre respondre un total de **89** preguntes en **10,28** minuts, explorant tot el contingut del graf. Si es calcula el temps mitjà de resposta, resulta en uns **6,93** segons per pregunta, la qual cosa mostra que, fins i tot amb un graf amb força interconnexions, el sistema és capaç de mantenir una velocitat de resposta adequada. Aquest resultat és especialment rellevant per a

escenaris pràctics, ja que un temps de resposta inferior als 10 segons per consulta sol ser considerat acceptable en entorns interactius.

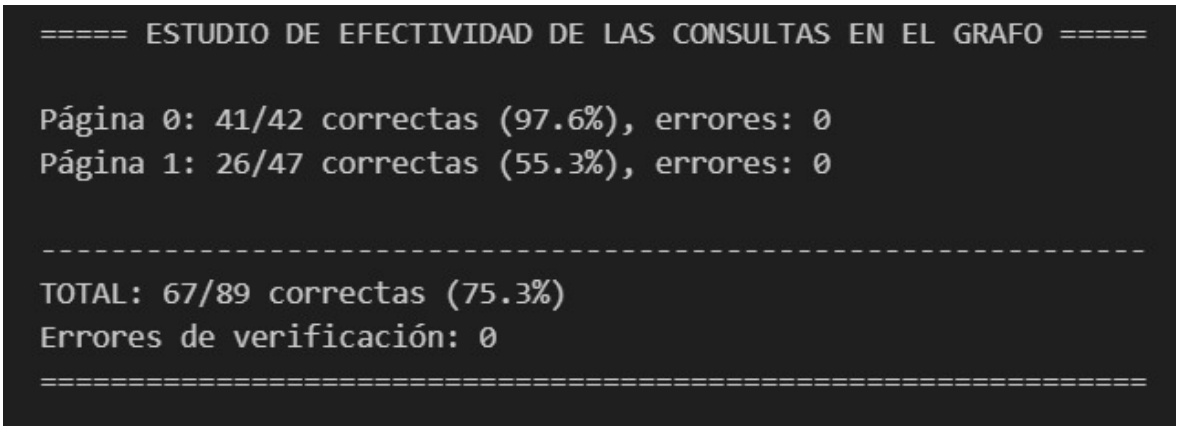


Figura 4. Resultats del joc de proves del currículum

Pel que fa a la prova amb un PDF de **49** pàgines corresponent a una regulació tècnica, els resultats posen de manifest la escalabilitat del sistema. En només **3,79 minuts** es va aconseguir construir i desar el graf complet a la base de dades Neo4j, gràcies al fet que les entitats, atributs i relacions ja havien estat verificats i estructurats prèviament en fitxers JSON. Aquesta eficiència és un indicador clau de la capacitat del sistema per tractar documents extensos en un temps raonable, sense comprometre la qualitat del coneixement representat.

A més, es va dur a terme una prova de rendiment específic en la fase de consulta interactiva, on es va variar el paràmetre corresponent al límit de sentències a explorar dins del graf. Aquest ajustament permet modular la profunditat de la cerca per optimitzar la relació entre temps de resposta i precisió de les respostes. Els resultats obtinguts van ser els següents:

• Amb un límit de **400 sentències,** es van respondre **132** preguntes en **28,50** minuts, amb **47** respostes incorrectes.

• Amb un límit de **200 sentències**, el temps total es va reduir a **22,40** minuts, amb **40** respostes incorrectes.

• Amb un límit de **50 sentències**, el temps de resposta va ser de només **16,27 minuts**, amb **32** respostes incorrectes.

Aquestes proves demostren que existeix un compromís natural entre velocitat i qualitat. Com més gran és el límit de sentències a explorar, més exhaustiva és la cerca, però també augmenta el temps de processament i la probabilitat d'obtenir respostes redundants o confuses. En canvi, amb límits més reduïts es guanya velocitat, però es pot perdre part de la cobertura. Així, el sistema ofereix la flexibilitat de configurar aquest paràmetre segons les necessitats concretes de l'usuari: consultes ràpides amb un marge d'error acceptable o bé respostes més fiables a canvi d'un temps de processament superior.

En conjunt, els resultats indiquen que el sistema és capaç de construir grafs de coneixement robustos i eficients tant en documents curts com llargs, i que permet resoldre preguntes en temps competitius amb uns nivells de precisió acceptables per a un entorn acadèmic i tècnic.

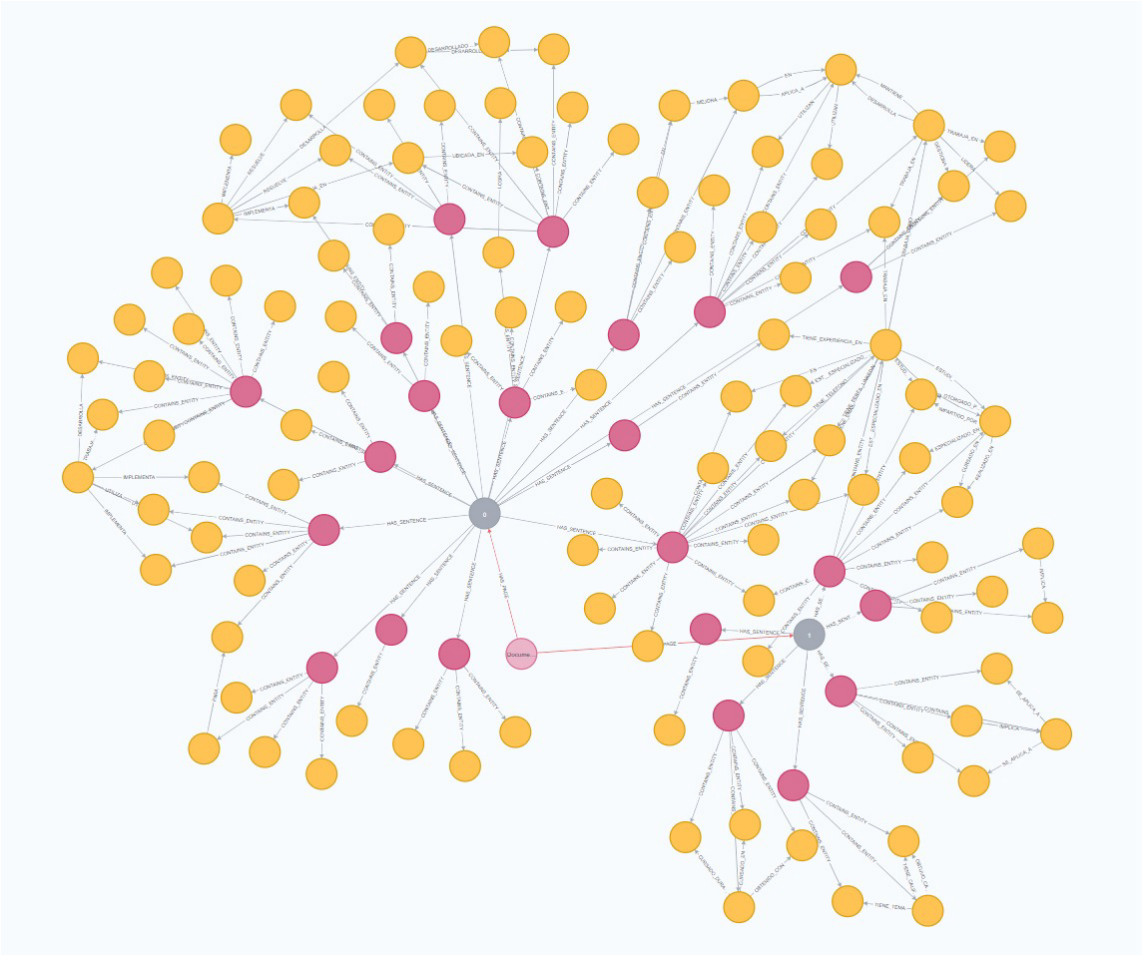


Figura 5. Graf en Neo4j del currículum complet

El graf representa l'estructura jeràrquica i semàntica d'un document, en aquest cas un currículum vitae. El punt de partida és el nodo arrel, que correspon al document complet i actua com a nucli central del qual es desplega tota la informació.

A partir d'aquest **nodo arrel (node rosa fluix)** es deriven les diferents **pàgines (node gris)** que conformen el document, reflectint així la seva naturalesa seqüencial. Cada pàgina es connecta amb les seves respectives **sentències (node rosa)**, que recullen fragments textuals més petits i granulars.

A dins de cada sentència es localitzen les **entitats (node groc)**, que són els conceptes, persones, organitzacions o elements clau identificats. Aquestes entitats disposen de metadades associades que enriqueixen la informació, com pot ser el tipus d'entitat, la seva categoria o el context en què apareixen.

Cal destacar que les entitats no només depenen de les sentències que les contenen, sinó que també poden establir relacions entre elles, creant una xarxa d'interconnexions que reflecteix millor la semàntica global del document. Això permet entendre no només el contingut de manera lineal (**pàgina → sentència → entitat**), sinó també les relacions conceptuals que existeixen entre diferents parts del currículum.

En aquest cas concret, el graf conté aproximadament **130 nodes i 185 relacions**, fet que mostra la riquesa estructural i semàntica del document. Aquesta representació no sols facilita la navegació i l'anàlisi del contingut, sinó que també ofereix una base sòlida per a

aplicacions posteriors, com ara consultes intel·ligents o processos d'enriquiment de coneixement.

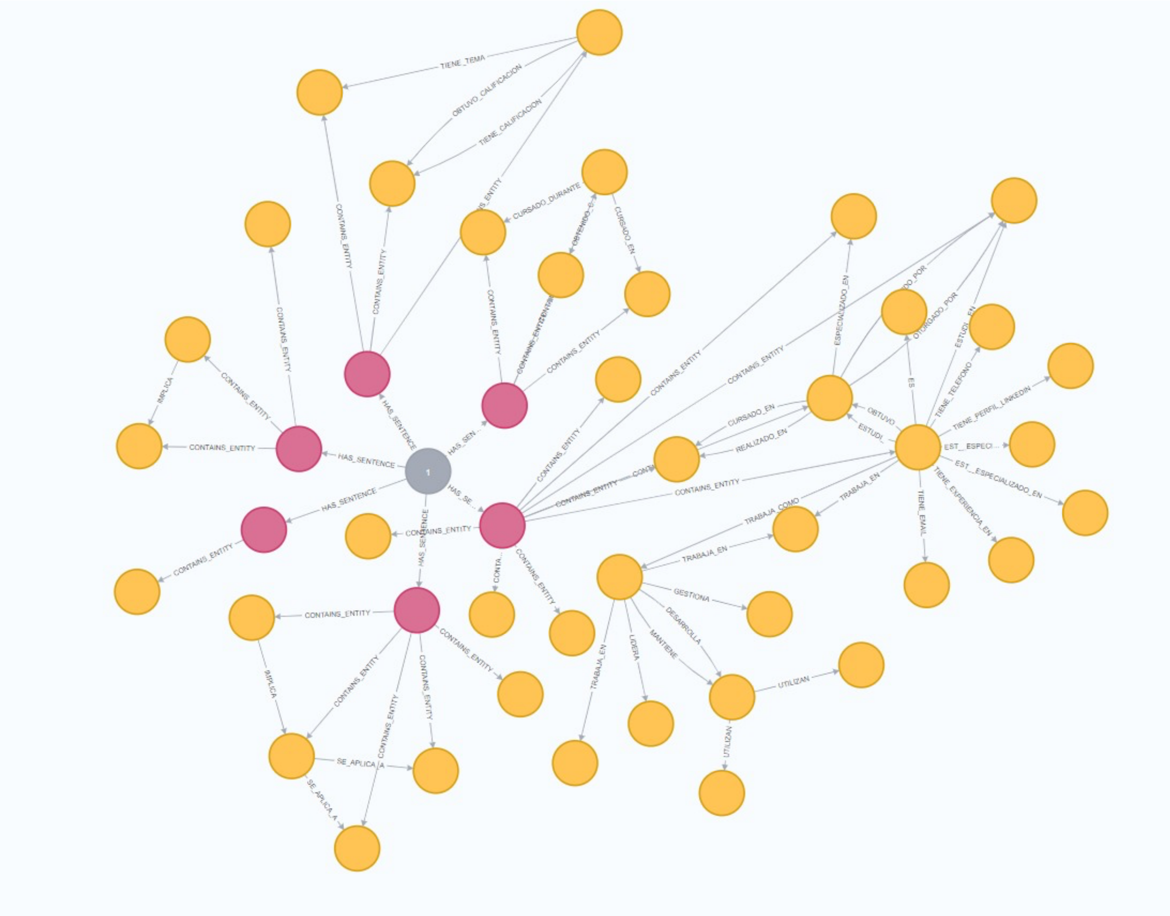


Figura 6. Graf en Neo4j de la primera pàgina del currículum

A nivell pràctic, per fer cerques i exploracions dins el graf de forma visual i interactiva, n'hi ha prou amb utilitzar Neo4j Browser, l'eina oficial que permet executar consultes en Cypher i visualitzar-ne els resultats en forma de xarxa. D'aquesta manera, es poden plantejar preguntes com ara: quins nodes té una pàgina concreta?, quines entitats apareixen dins d'ella? o quins són els veïns d'una entitat i amb qui té relació?.

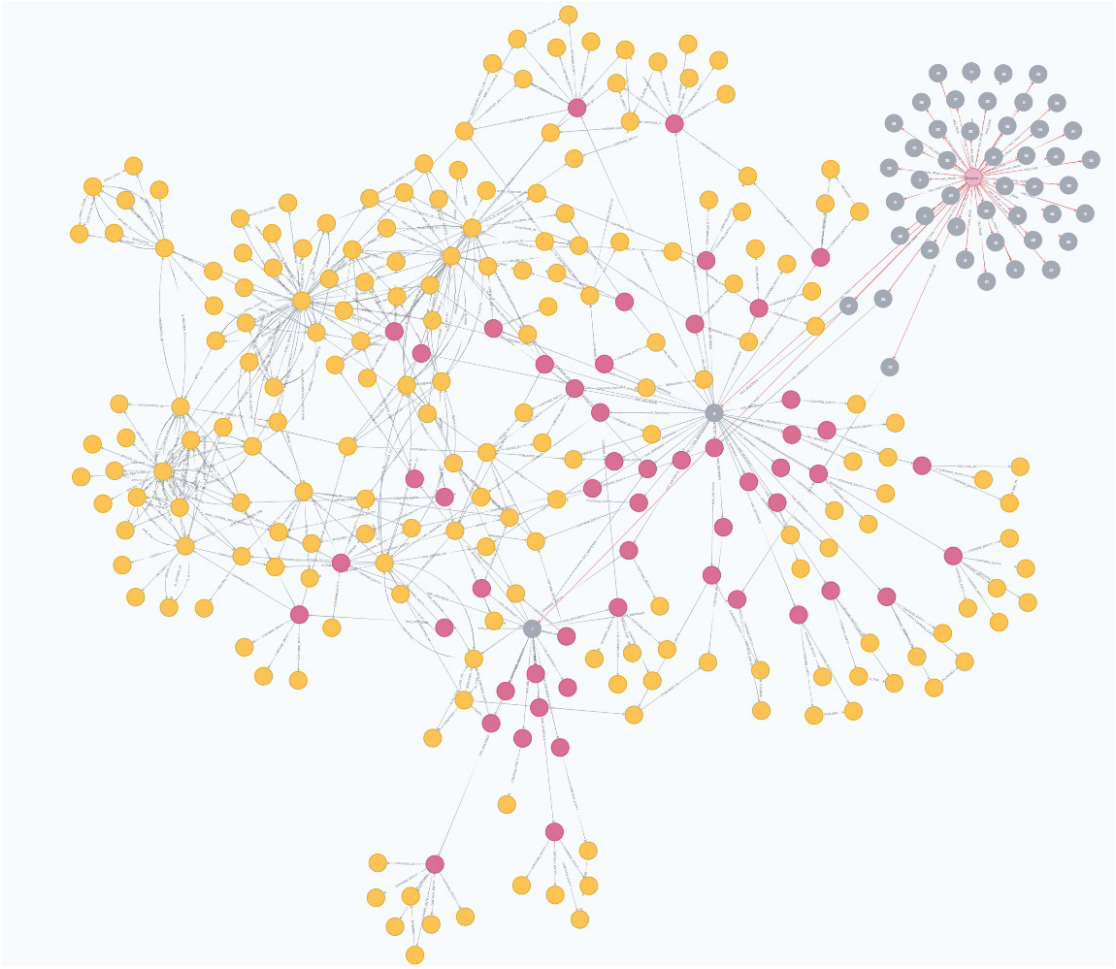


Figura 7. Part del graf de 49 pàgines a Neo4j

A més del Neo4j Browser, també es poden fer cerques mitjançant Neo4j Bloom (que ofereix una interfície més intuïtiva i orientada a usuaris no tècnics), o bé connectar el a eines externes de visualització i anàlisi com Graph Data Science, Gephi o llibreries de Python com NetworkX i PyVis, que permeten generar visualitzacions més personalitzades i aplicar algorismes avançats d'anàlisi de xarxes.

8.4 Conclusions de l'avaluació

Les proves realitzades demostren que:

• El sistema és funcional, robust i traçable.

• L'ús del LLM per verificar millora significativament la qualitat del coneixement final.

• La representació jeràrquica al graf facilita consultes eficients i modulars.

• El sistema encara pot millorar en cobertura semàntica (ex: frases amb metàfores o informació no explícita).

9 Costos del projecte

En aquest apartat es detallen els costos associats al desenvolupament del projecte, tenint en compte el temps invertit, els recursos humans i els recursos materials i tecnològics utilitzats.

9.1 Costos en temps

El projecte s'ha desenvolupat al llarg d'aquest estiu de 2025 amb una dedicació mitjana estimada de **20** hores setmanals. Tot i això, durant les fases finals de desenvolupament i validació la intensitat de treball es va incrementar, arribant a dedicar més hores de manera concentrada per tal de garantir la qualitat i la finalització del sistema dins del calendari previst.

Un element clau en la gestió del temps i en el progrés del projecte ha estat la celebració de **reunions setmanals** cada dijous amb el tutor de l'empresa. Aquestes trobades han tingut com a objectiu:

• Posar al dia l'evolució del projecte, comentant les tasques realitzades durant la setmana.

• Rebre orientació i feedback sobre les decisions tècniques preses i els resultats obtinguts.

• Definir els propers passos a curt termini, establint petits objectius setmanals que permetien avançar de manera constant.

• Resoldre dubtes immediats i anticipar possibles dificultats tècniques o de planificació.

Aquestes reunions han estat molt útils no només per mantenir un ritme de treball estable, sinó també per assegurar que el projecte es desenvolupés alineat amb les expectatives de l'empresa. El fet de poder compartir setmanalment els avenços va permetre identificar ràpidament possibles desviacions, corregir-les i establir estratègies d'optimització.

A nivell personal, aquesta dinàmica va contribuir a millorar l'organització del temps i a desenvolupar una disciplina de treball per objectius, ja que cada setmana hi havia un compromís de presentar progressos tangibles.

9.2 Recursos humans

El projecte ha estat desenvolupat íntegrament per una estudiant de grau sota la tutoria acadèmica i la supervisió del tutor d'empresa. La col·laboració amb l'equip **d'IDIADA** ha aportat orientació puntual sobre l'ús de recursos interns, com ara el perfil **aida-dev** per accedir a serveis de computació i LLM via **AWS Bedrock**.

Tot i que el desenvolupament ha estat individual, la validació del projecte ha comptat amb feedback tècnic del responsable de pràctiques dins l'empresa.

9.3 Recursos materials i tecnològics

Els recursos materials utilitzats durant la realització del treball de final de grau han estat mínims i majoritàriament d'accés obert, excepte l'ús de la infraestructura en núvol proporcionada per l'empresa **IDIADA**.

Concretament, la empresa ha facilitat l'accés a la plataforma Amazon Web Services (AWS) mitjançant el perfil corporatiu aida-dev. Aquest recurs ha estat clau per accedir a models d'intel·ligència artificial avançats, com el model Anthropic Claude 3.5 Sonnet, a través del servei **AWS Bedrock**, que ha permès realitzar la generació de respostes i la verificació basada en **LLM**.

Pel que fa a **l'equip físic**, l'ordinador utilitzat per al desenvolupament ha estat proporcionat per l'empresa, però el programari i les eines de desenvolupament han estat majoritàriament lliures i de codi obert, utilitzant tecnologies com Python, spaCy, i la base de dades de graf Neo4j Community Edition.

També s'ha fet ús de biblioteques i serveis en el núvol amb plans gratuïts o d'ús públic, com ara Amazon Comprehend per a l'extracció d'entitats i relacions, i entorns de programació estàndard, sense necessitat de licències específiques.

10 Protecció de dades i legislació

Durant el desenvolupament d'aquest projecte **no s'han tractat dades personals** en cap moment. Tots els documents utilitzats corresponen a exemples genèrics o regulacions típiques pròpies de l'entorn empresarial, sense cap informació sensible o identificable de persones físiques.

Així doncs, no ha estat necessari aplicar mesures específiques per a la protecció de dades personals segons el Reglament General de Protecció de Dades (RGPD). No obstant això, s'ha mantingut un compromís estricte amb les bones pràctiques en el tractament de la informació, garantint la confidencialitat i la seguretat dels documents d'exemple processats.

El projecte s'ha desenvolupat utilitzant la infraestructura de l'empresa a AWS, la qual ofereix garanties de seguretat i compliment normatiu, assegurant que la informació, encara que no contingui dades personals, es mantingui en un entorn segur i controlat, amb accés limitat als professionals implicats en el projecte.

A més, s'ha tingut especial cura a respectar la legislació relativa a la propietat intel·lectual i drets d'autor sobre els documents utilitzats, garantint que el seu ús es limita exclusivament a finalitats educatives i de recerca dins del marc del projecte i la política de l'empresa.

11 Implicacions ètiques, ambientals i d'igualtat

El projecte desenvolupat presenta diverses implicacions que cal considerar des d'una perspectiva ètica, ambiental i d'igualtat, tant pel que fa a la seva concepció, implementació i possibles usos futurs.

11.1 Implicacions ètiques

En primer lloc, des del punt de vista ètic, és fonamental destacar que el sistema implementat no tracta dades personals ni informació sensible, evitant així riscos directes relacionats amb la privacitat i la protecció de dades dels individus.

Tanmateix, la naturalesa del projecte, que implica la manipulació i anàlisi automàtica de textos i documents, implica una responsabilitat en assegurar que les decisions i respostes generades per l'agent basat en models d'intel·ligència artificial siguin **fiables**, no enganyoses ni discriminatòries.

Per tant, s'ha incorporat un **agent verificador** basat en un model de llenguatge gran (LLM) que avalua i valida la qualitat de les entitats, atributs i relacions extretes, minimitzant errors i possibles biaixos que podrien afectar la interpretació del coneixement. Aquest enfocament contribueix a promoure la transparència, responsabilitat i la fiabilitat de la informació processada.

A més, és important considerar que qualsevol aplicació futura d'aquesta tecnologia hauria d'estar subjecta a una avaluació ètica rigorosa, especialment si s'utilitza en contextos on les decisions automàtiques puguin tenir un impacte significatiu sobre persones o organitzacions.

11.2 Implicacions ambientals

En relació amb l'impacte ambiental, cal tenir en compte que l'ús d'infraestructures d'intel·ligència artificial basades en núvol, com la plataforma AWS utilitzada per a l'entrenament i execució dels models, consumeix **recursos energètics** importants. Tot i que es tracta d'un projecte de prova de concepte i investigació amb un ús moderat, la conscienciació sobre l'eficiència energètica és clau.

La infraestructura cloud utilitzada per la empresa IDIADA, així com les bones pràctiques recomanades per AWS, inclouen esforços per millorar la sostenibilitat energètica, com l'ús d'energia renovable i optimitzacions en l'ús de recursos. És responsabilitat dels desenvolupadors i les organitzacions monitoritzar i minimitzar la petjada ambiental associada a l'ús d'aquestes tecnologies, promovent un ús responsable i sostenible.

Aquest projecte posa de manifest la necessitat de considerar, en futures implementacions i escales més grans, estratègies d'optimització per reduir l'ús **computacional** i, per tant, l'impacte **mediambiental**, com ara **l'optimització dels model**s, la **reducció de consultes** innecessàries i la **reutilització de càlculs previs**.

11.3 Implicacions d'igualtat

Pel que fa a la dimensió d'igualtat, el projecte promou la igualtat d'accés a la informació i el coneixement, permetent que tècnics i enginyers puguin recuperar informació rellevant de manera eficient i accessible a través d'un sistema automatitzat.

No obstant això, cal ser conscients que els models d'intel·ligència artificial poden presentar biaixos inherents derivats de les dades amb què han estat entrenats. Aquests biaixos poden traduir-se en una representació desigual o discriminatòria en la recuperació i presentació d'informació. Per aquest motiu, la inclusió d'un agent verificador que avalua i enriqueix el coneixement extret contribueix a mitigar possibles desequilibris.

És important també que la interfície i l'ús de l'eina siguin accessibles per a usuaris amb diferents capacitats i procedències, evitant barreres tecnològiques o culturals que puguin excloure certs col·lectius. La promoció d'una experiència d'usuari inclusiva i l'adequació a normes d'accessibilitat són aspectes que es recomanen considerar en futures fases de desenvolupament.

12 Conclusions

12.1 Avaluació de l'aportació del TFG a la formació personal

La realització d'aquest Treball de Fi de Grau ha suposat un repte significatiu que ha permès consolidar i ampliar els coneixements adquirits al llarg de la formació acadèmica, especialment en l'àmbit del processament de llenguatge natural, l'extracció de coneixement automàtica i la interacció amb sistemes basats en intel·ligència artificial.

Aquest projecte ha afavorit un aprenentatge pràctic molt valuós, en el qual s'han integrat múltiples disciplines com la programació avançada en Python, la manipulació de dades semiestructurades, l'ús d'APIs de serveis cloud (AWS Bedrock, Comprehend), i el treball amb bases de dades de graf (Neo4j). Això ha permès aprofundir en tecnologies emergents i preparar-se per afrontar reptes reals en entorns industrials i empresarials.

També s'ha desenvolupat habilitats de disseny i implementació d'arquitectures de sistemes, així com la capacitat de planificació i gestió d'un projecte complex, des del disseny de requisits fins a la validació i avaluació dels resultats. La integració de models de llenguatge per a la verificació i enriquiment del coneixement extret ha estat una experiència enriquidora que obre la porta a futures línies d'investigació i aplicació professional.

A nivell personal, aquest treball ha fomentat la iniciativa, la resolució de problemes i la perseverança davant dels obstacles tècnics i conceptuals que s'han anat trobant al llarg del procés. L'experiència de col·laborar amb l'empresa IDIADA ha aportat una visió pràctica i aplicada que complementa la formació teòrica rebuda a la universitat.

12.2 Valoració personal

La valoració personal d'aquest projecte és molt positiva. La complexitat i transversalitat dels temes abordats han suposat un estímul intel·lectual important i han permès experimentar amb tecnologies punteres en intel·ligència artificial i processament del llenguatge natural. La capacitat d'autogestió i aprenentatge continuat ha estat clau per a l'èxit de la iniciativa.

L'aprenentatge sobre la interacció amb models LLM al núvol, la gestió de dades semiestructurades i l'optimització del procés de recuperació de coneixement són competències que consider que em seran molt útils en la meva carrera professional. També valoro positivament la implementació d'un sistema robust que integra múltiples components, així com el desenvolupament de prompts i agents específics per a la verificació de qualitat, que representa un pas innovador i diferenciador.

En un altre nivell, l'experiència ha reforçat la importància de considerar aspectes ètics, de sostenibilitat i d'igualtat en el desenvolupament tecnològic, fet que considero essencial per al professional actual i futur. La pràctica amb entorns cloud i la col·laboració amb un equip professional han enriquit molt la meva perspectiva i preparació.

12.3 Millores i Treball Futur

Tot i que el sistema actual implementa un flux complet de processament, verificació i consulta sobre un graf de coneixement a partir de documents no estructurats, existeixen diversos àmbits de millora que poden incrementar la flexibilitat, escalabilitat i eficiència de la solució:

• Gestió dinàmica del graf de coneixement (CRUD):

Actualment el graf s'alimenta a partir d'un procés seqüencial d'extracció, verificació i importació. Una millora clau seria habilitar operacions de creació, lectura, actualització i eliminació (CRUD) sobre entitats, relacions i atributs de manera interactiva. D'aquesta forma, els usuaris podrien modificar el graf quan fos necessari i les actualitzacions es propagarien automàticament al sistema, mantenint la base de coneixement actualitzada sense necessitat de repetir tot el pipeline.

• Ampliació de la jerarquia de nodes amb imatges i taules:

El model actual contempla documents, pàgines, sentències, entitats, relacions i atributs. Una millora significativa seria afegir un nou nivell específic per a imatges i taules contingudes en els documents. Això permetria representar gràficament no només el contingut textual sinó també el visual i estructurat. Actualment, el sistema ja identifica el contingut de les taules i, si es modifica el paràmetre extract_image al mòdul PDF Processor, és capaç de detectar també el contingut d'imatges. Formalitzar aquesta informació dins del graf aportaria més riquesa semàntica i completaria la representació del coneixement.

• Optimització de temps en la verificació i exportació:

El procés de verificació i exportació dels JSON verificats pot resultar costós en termes de temps quan el volum de dades és elevat. Una millora seria introduir mecanismes de processament paral·lel i asincrònic, de manera que la verificació de pàgines o seccions es pugui fer en paral·lel, reduint significativament els temps d'execució totals.

• Automatització de paràmetres del pipeline (idioma i límit de sentències):

Actualment, el pipeline requereix definir manualment paràmetres com el llenguatge de processament i el nombre màxim de sentències a verificar per lot (limit_sentences). Una millora seria implementar un agent de detecció automàtica capaç de seleccionar l'idioma del document i ajustar dinàmicament el límit de sentències en funció de la longitud i la densitat del text. Això permetria reduir la intervenció manual i augmentar la robustesa del sistema davant documents heterogenis.

En conjunt, aquestes millores apunten a convertir el sistema en una eina més interactiva, multimodal i eficient, capaç de gestionar coneixement de forma més flexible i autònoma, i que obre camí a futures aplicacions en entorns corporatius o acadèmics on la gestió de grans volums d'informació és crítica.

12.4 Assoliment dels objectius

Els objectius plantejats al principi del projecte s'han complert de manera satisfactòria, tal i com es reflecteix en els resultats obtinguts. En concret:

• S'ha desenvolupat un sistema capaç d'extreure automàticament entitats, atributs i relacions de documents textuals en format PDF, amb un processament per pàgines que facilita la gestió i validació de la informació.

• S'ha implementat un agent basat en models LLM que avalua i enriqueix la informació extreta, millorant la precisió i la consistència del coneixement representat.

• S'ha integrat una arquitectura flexible que combina serveis cloud (AWS Bedrock), bases de dades de graf (Neo4j) i eines de processament lingüístic (spaCy, Comprehend), garantint escalabilitat i mantenibilitat.

• S'ha documentat i avaluat rigorosament la qualitat del coneixement extret mitjançant la definició de casos d'ús i l'execució de proves que asseguren la robustesa del sistema.

• S'han tingut en compte aspectes transversals com la seguretat de dades, l'ètica i la sostenibilitat, donant un enfocament integral al projecte.

No obstant això, s'identifiquen àrees de millora i possibles extensions que es podrien abordar en treballs futurs, com ara una major automatització del procés d'enriquiment, la integració d'altres fonts de dades, o la millora dels algoritmes de recuperació i resposta per a contextos més complexos.

En conclusió, aquest TFG ha estat una experiència de gran valor acadèmic i professional que ha permès assolir amb èxit els objectius plantejats, i que proporciona una base sòlida per al desenvolupament futur tant a nivell tècnic com conceptual.

13 Referències

[1] Goodfellow, I., Bengio, Y., Courville, A., Deep Learning, MIT Press, 2016.

[2] Lewis, P., Perez, E., Piktus, A., et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Advances in Neural Information Processing Systems (NeurIPS), 2020, Vol. 33, pp. 9459–9474.

[3] Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S., A Survey on Knowledge Graphs: Representation, Acquisition, and Applications, IEEE Transactions on Neural Networks and Learning Systems, 2022, Vol. 33, No. 2, pp. 494–514.

[4] Lewis, P., et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Advances in Neural Information Processing Systems, 2020.

[5] Dean, J., & Ghemawat, S., MapReduce: Simplified Data Processing on Large Clusters, OSDI, 2004.

[6] Vakulenko, S., et al., A Survey on Natural Language Query Interfaces to Knowledge Bases, Semantic Web Journal, 2019.

[7] Hogan, A., et al., Knowledge Graphs, Synthesis Lectures on Data, Semantics, and Knowledge, Morgan & Claypool, 2021.

[8] Francis, N., et al., Cypher: An Evolving Query Language for Property Graphs, SIGMOD Record, 2018.

[9] Lewis, P., et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Advances in Neural Information Processing Systems, 2020.

[10] Gao, L., et al., Precise Zero-Shot Dense Retrieval without Relevance Labels, arXiv preprint arXiv:2109.01794, 2021.

[11] Amazon Web Services, Amazon Bedrock – Foundation Model APIs, https://aws.amazon.com/bedrock [consulta: 31/08/2025].

[12] Manning, C., & Schütze, H., Foundations of Statistical Natural Language Processing, MIT Press, 1999.

[13] Cambria, E., et al., Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems, O'Reilly, 2020.

[14] Hochreiter, S., & Schmidhuber, J., Long Short-Term Memory, Neural Computation, 1997.

[15] Smith, R., An Overview of the Tesseract OCR Engine, Document Recognition and Retrieval XVII, SPIE, 2007.

[16] Pàgina web: AWS Comprehend, Documentación de Amazon Comprehend, https://aws.amazon.com/comprehend/. Consulta 31/08/2025.

[17] Llibre: Samuel Albanie, Pydantic: Data validation and settings management using Python type annotations, O'Reilly Media, 2023.

[18] Article de Revista: José García, María López, "Evaluación de la calidad de datos en grafos de conocimiento: metodologías y métricas", Revista Española de Inteligencia Artificial, 2021, Vol. 25, Núm. 70, pp. 35-48.

[19] Article de Revista: Antonio S. Ruiz, Laura Martín, "Uso de técnicas de validación semántica en sistemas de extracción de información", Inteligencia Artificial. Revista Iberoamericana de IA, 2020, Vol. 23, Núm. 65, pp. 15-28.

[20] Article de Revista: Carmen Delgado, Javier Torres, "Métricas de precisión y calidad en la verificación de conocimiento extraído", Revista Española de Documentación Científica, 2022, Vol. 45, Núm. 2, pp. 1-12.

[21] Angles, R., & Barceló, P. Graph Databases: Models, Languages, and Systems. Morgan & Claypool Publishers, 2017.

[22] Domínguez-Sal, D., Martínez-Bazan, N., & Larriba-Pey, J. L. A discussion on the design of graph database benchmarks. Universitat Politècnica de Catalunya, 2010.

[23] Neo4j. The Neo4j Graph Platform. Neo4j, 2023. Disponible a: HTTP://neo4j.com [consulta: 01/09/2025].

[24] Py2neo Documentation. Py2neo v2021.2.3. Disponible a: HTTP://py2neo.org [consulta: 01/09/2025].

[25] Larriba-Pey, J. L., Domínguez-Sal, D., Martínez-Bazan, N. Graph data management: techniques and applications in Spain. Revista Española de Informática, 2015, Vol. 12, Núm. 2, p. 45–60.

[26] Hogan, Aidan, et al. Knowledge Graphs. ACM Computing Surveys, 2021, Vol. 54, Núm. 4, p. 1–37.

[27] Suchanek, F. M., Kasneci, G., & Weikum, G. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. WWW Conference, 2007.

[28] Chaves-Fraga, D., Priyatna, F., Corcho, O. Enhancing Data Quality in Knowledge Graphs. Revista de la Sociedad Española de Procesamiento de Lenguaje Natural, 2020, Núm. 64, p. 13–22.

[29] Llibre: Robinson, I., Webber, J., & Eifrem, E. Graph Databases. O'Reilly Media, 2015.

[30] Article de Revista: Antolí, A., Martínez-Romo, J., & Araujo, L. Análisis semántico y verificación de [5] Article de Revista: Chen, W., Chen, X., & Yu, K. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks: A Survey. arXiv, 2023.

[31] Article de Revista: Yao, S., Yu, D., Chen, J., Weng, J., Zhao, Z., & Narasimhan, K. ReAct: Synergizing Reasoning and Acting in Language Models. ICLR, 2023.

[32] Article de Revista: García, J., "Tendencias en el uso de modelos de lenguaje en entornos cloud", Revista Española de Inteligencia Artificial, 2023, Vol. 27, Núm. 101, p. 55-68.

[33] Article de Revista: López, A.; Sánchez, M., "Evaluación de trade-offs en modelos de lenguaje para NLP", Novática, 2022, Vol. 249, Núm. 3, p. 20-29.

[34] Pàgina Web: https://aws.amazon.com/bedrock.

[35] Article de Revista: Ferrer, D.; Martín, P., "Validación de datos en NLP con Pydantic", Revista Iberoamericana de Computación, 2024, Vol. 12, Núm. 2, p. 77-85.

[36] Llibre: O'Reilly, "Learning JSON", O'Reilly Media, 2020.

[37] Llibre: Robinson, I.; Webber, J.; Eifrem, E., "Graph Databases", O'Reilly Media, 2015.

[38] Article de Revista: González, R., "Uso de Cypher para la consulta eficiente de grafos", RIAI - Revista Iberoamericana de Automática e Informática Industrial, 2021, Vol. 18, Núm. 4, p. 329-337.

[39] Amazon Web Services. Amazon Comprehend – Natural Language Processing (NLP) Service. AWS, 2024.

[40] Montani, I., & Honnibal, M. spaCy: Industrial-Strength Natural Language Processing in Python. Explosion AI, 2023.

[41] García-Silvente, M., & Valls, A. Sistemas Inteligentes: Fundamentos y Aplicaciones. Editorial UOC, 2020.

[42] Robinson, J., & Webber, J. Graph Databases: New Opportunities for Connected Data. O'Reilly Media, 2018.

[43] Domingo-Ferrer, J. Privacidad y trazabilidad en sistemas basados en grafos de conocimiento. Revista Española de Documentación Científica, 2022, Vol. 45, Núm. 2, pp. 1–14.

[44] Amazon Web Services. Amazon Bedrock – Machine Learning Service for Large Language Models. AWS, 2024.

[45] Anthropic. Claude 3.5 Sonnet – Documentation for Developers. Anthropic & AWS, 2024.

[46] Robinson, J., & Webber, J. Graph Databases: New Opportunities for Connected Data. O'Reilly Media, 2018.

14 Annexes

14.1 Exemples de consultes Cypher

1. Visualització de tot un graf

MATCH (n)-[r]-(m)

RETURN n, r, m

2. Visualització de la primera pàgina del document

MATCH path = (root {id: 'page_1'})-[*0..]->()

RETURN path

ORDER BY length(path)

14.2 Dependències

- Python 3.10
- Biblioteques: boto3, spacy, py2neo, pydantic, concurrent.futures, pandas, json
- Neo4j 5.x amb accés local o remot
- Accés a AWS Bedrock amb perfil aida-dev

14.3 Codis

14.3.1 Creació client bedrock

```
bedrock_client = boto3.Session(profile_name="aida-dev").client(
    "bedrock-runtime", region_name="us-east-1"
)
```

14.3.2 Inicialitzar Comprehend i Spacy (en espanyol)

```
comprehend = boto3.Session(profile_name='aida-dev').client('comprehend',
region_name='us-west-2')

nlp = spacy.load("es_core_news_md")
```

14.3.3 Extracció entitats

```
def extract_entities(text):
    response = comprehend.detect_entities(
        Text=text,
        LanguageCode='es'
    )
    entities = response.get('Entities', [])
    filtered_entities = [e for e in entities if e.get('Score', 0) > 0.1]
    return filtered_entities
```

14.3.4 Creació agent per afegir, modificar i evaluar elements

```
def create_knowledge_quality_checker_agent(language="es"):
    """
    Crea un agente para verificar la calidad de las entidades, atributos y
relaciones extraídos de un texto,
    con capacidad de añadir, modificar o completar elementos faltantes.
```

```
    Args:
        language: El idioma en el que se realizará la verificación (default: "es"
para español)

    Returns:
        Agent: Un agente configurado para verificación y enriquecimiento completo
    """

    # Configuració del LLM
    model_id = "anthropic.claude-3-5-sonnet-20240620-v1:0"
    aws_session = boto3.Session(profile_name='aida-dev', region_name="us-west-2")
    llm_config = AWSLLMConfig(session=aws_session, llm_model_id=model_id)

    system_prompt = f"""
    Eres un experto en procesamiento de lenguaje natural y extracción de
conocimiento en {language}.
    Tu tarea consiste en evaluar, corregir y ENRIQUECER las entidades, atributos
y relaciones que han sido extraídos automáticamente de un texto.

    **FUNCIONES PRINCIPALES:**

    1. **EVALUACIÓN Y CORRECCIÓN:**
       - Evaluar si el tipo de entidad asignado es correcto según el contexto del
texto completo
       - Corregir clasificaciones erróneas de entidades
       - Verificar y ajustar la delimitación del texto de las entidades
       - Corregir puntuaciones de confianza incoherentes

    2. **ENRIQUECIMIENTO Y ADICIÓN:**
       - **AÑADIR entidades importantes que no fueron detectadas** en el texto
original
       - **COMPLETAR atributos faltantes** para entidades existentes
       - **CREAR nuevas relaciones** que sean evidentes en el texto pero no
fueron extraídas
       - **MEJORAR la granularidad** de entidades demasiado generales

    3. **VALIDACIÓN CONTEXTUAL:**
       - Asegurar coherencia semántica entre todos los elementos
       - Verificar que las relaciones sean lógicamente consistentes
       - Completar cadenas de relaciones implícitas en el texto

    **CRITERIOS DE ENRIQUECIMIENTO:**

    ### ENTIDADES - Añadir si faltan:
    - PERSON: Nombres de personas mencionadas pero no extraídas
    - LOCATION: Lugares geográficos implícitos o explícitos
    - ORGANIZATION: Empresas, instituciones, agencias no detectadas
    - DATE: Fechas o períodos temporales mencionados
    - QUANTITY: Números, medidas, cantidades específicas
    - EVENT: Eventos o acontecimientos relevantes
```

```
    - TITLE: Títulos de obras, documentos, proyectos
    - COMMERCIAL_ITEM: Productos, servicios, tecnologías
    - FACILITY: Instalaciones, edificios, infraestructuras
    - MATERIAL: Materiales, sustancias, compuestos químicos
    - CONCEPT: Conceptos abstractos importantes
    - OTHER: Otras entidades relevantes no categorizadas

    ### ATRIBUTOS - Completar información:
    - Propiedades físicas (tamaño, color, forma)
    - Características temporales (duración, frecuencia)
    - Propiedades cualitativas (calidad, estado, condición)
    - Relaciones de pertenencia o asociación
    - Características cuantitativas (cantidad, porcentaje, medidas)

    ### RELACIONES - Crear conexiones faltantes (SIN NINGÚN ACENTO):
    - Relaciones de causa-efecto evidentes en el texto
    - Relaciones temporales (antes, después, durante)
    - Relaciones espaciales (en, cerca de, dentro de)
    - Relaciones jerárquicas (parte de, pertenece a)
    - Relaciones funcionales (utiliza, produce, controla)

    **INSTRUCCIONES ESPECÍFICAS:**
    - Sé **proactivo** en la identificación de elementos faltantes
    - **Mantén alta precisión** - solo añade elementos que estén claramente
respaldados por el texto
    - **Preserva el contexto** - asegúrate de que las adiciones sean coherentes
con el tema general
    - **Prioriza la relevancia** - enfócate en elementos que aporten valor al
conocimiento extraído
    - **Marca claramente** los elementos que has añadido vs. los que has
corregido
    """

    prompt_template = """
    Por favor, evalúa, corrige y ENRIQUECE el conocimiento extraído del siguiente
texto:

    ### TEXTO ORIGINAL:
    ```
    {text}
    ```

    ### ELEMENTOS EXTRAÍDOS INICIALMENTE:

    #### ENTIDADES:
    ```
    {entities}
    ```

    #### ATRIBUTOS:
    ```
```
```

```
    {attributes}
    ```

    #### RELACIONES:
    ```
    {relations}
    ```

    ### TU TAREA COMPLETA:

    1. **EVALÚA** cada elemento extraído para identificar errores o imprecisiones
    2. **CORRIGE** cualquier clasificación errónea o delimitación incorrecta
    3. **IDENTIFICA** entidades importantes que no fueron detectadas en el texto
    4. **AÑADE** atributos relevantes que faltan para las entidades existentes y
nuevas
    5. **CREA** relaciones adicionales que sean evidentes en el texto pero no
fueron extraídas
    6. **MEJORA** la granularidad y precisión de todos los elementos

    ### FORMATO DE RESPUESTA:
    Para cada sección, indica claramente:
    - ✅ **CORRECCIONES**: Elementos que has corregido (explica qué cambió)
    - ➕ **ADICIONES**: Nuevos elementos que has añadido (explica por qué)
    - 🔍 **VALIDACIONES**: Elementos originales que confirmas como correctos

    IMPORTANTE: Solo añade elementos que estén **claramente respaldados** por
el texto original.
    Mantén un equilibrio entre exhaustividad y precisión.

    Utiliza la herramienta de formato para proporcionar tu evaluación y
enriquecimiento detallado.
    """

    # Configuració de l'eina de format
    formatter_tool_config = FormatterTool.get_formatter_tool_config(
        output_schema=KnowledgeQualityOutputSchema
)

    # Creació del agent
    knowledge_quality_agent = Agent(
        llm_config=llm_config,
        system_prompt=system_prompt,
        prompt_template=prompt_template,
        tool_config=formatter_tool_config,
        name="Enhanced Knowledge Quality & Enrichment Agent",
        description="Agente que evalúa, corrige y enriquece el conocimiento
extraído de un texto"
)

    return knowledge_quality_agent
```
```

14.3.5 Recuperació de context per la consulta de graf

```
def retrieve_context(keywords: List[str], user_query: str = "", limit_sentences:
int = 50) -> List[Dict[str, Any]]:
    """
    Recupera contexto del grafo, filtrando por número de punto/sección si la
pregunta lo menciona.
    """
    if not keywords:
        return []
    keywords = [normalize_text(kw) for kw in keywords]
    text_patterns = [f"toLower(s.text) CONTAINS '{kw}'" for kw in keywords]
    entity_patterns = [f"toLower(e1.text) CONTAINS '{kw}'" for kw in keywords]
    rel_patterns = [f"toLower(e2.text) CONTAINS '{kw}'" for kw in keywords]
    rel2_patterns = [f"toLower(e3.text) CONTAINS '{kw}'" for kw in keywords]
    rel_type_patterns = [f"toLower(type(r1)) CONTAINS '{kw}' OR toLower(type(r2))
CONTAINS '{kw}'" for kw in keywords]
    where_clause = " OR ".join(text_patterns + entity_patterns + rel_patterns +
rel2_patterns + rel_type_patterns)

    # Detectar si la pregunta menciona un punto/sección concreto
    section_number = extract_section_number(user_query)
    section_filter = ""
    if section_number:
        # Busca el número entre paréntesis al inicio de la oración, o en
cualquier parte si es necesario
        section_filter = f"AND s.text =~
'\\\\(\\\\s*{section_number}\\\\s*\\\\).*'"

    cypher = f"""
    MATCH (s:Sentence)-[:CONTAINS_ENTITY]->(e1:Entity)
    OPTIONAL MATCH (e1)-[r1]->(e2:Entity)
    OPTIONAL MATCH (e2)-[r2]->(e3:Entity)
    WHERE ({where_clause}) {section_filter}
    RETURN s.text AS sentence,
           collect(DISTINCT e1.text) + collect(DISTINCT e2.text) +
collect(DISTINCT e3.text) AS entities,
           collect(DISTINCT {{
             predicate: coalesce(type(r1), type(r2)),
             subject : coalesce(e1.text, e2.text),
             object  : coalesce(e2.text, e3.text)
           }}) AS relations
    LIMIT $limit
    """
    context = graph.run(cypher, limit=limit_sentences).data()
    logging.info(f"📄 Se recuperaron {len(context)} sentencias del grafo (filtro
sección: {section_number or 'ninguno'})")
    return context
```

14.3.6 Creació agent per verificar i evaluar els resultats de les consultes

```
def create_answer_verification_agent(language="en"):
    """
```

```
    Crea un agente LLM para verificar si la respuesta a una pregunta es correcta
según SOLO el texto de la página.
    """
    model_id = "anthropic.claude-3-5-sonnet-20240620-v1:0"
    aws_session = boto3.Session(profile_name='aida-dev', region_name="us-west-2")
    llm_config = AWSLLMConfig(session=aws_session, llm_model_id=model_id)

    system_prompt = (
        "Eres un verificador experto. Analiza si la respuesta a la pregunta es
correcta según el texto. "
        "Busca exhaustivamente en TODO el texto. Si la información está presente,
cítala literalmente como evidencia. "
        "Si la respuesta es general pero correcta (por ejemplo, 'motor vehicles'
cuando el texto dice 'motor vehicles'), acéptala como correcta. "
        "Solo responde 'no hay información' si tras buscar en todo el texto no
existe ninguna mención relevante. "
        "Devuelve SOLO un JSON con estos campos: is_correct (true/false),
justification (explicación), evidence (fragmento literal del texto usado)."
        "REGLA CLAVE: Si el LLM responde 'no hay información' o 'no se encontró'
pero la información SÍ existe en el texto usando sinónimos o palabras
equivalentes, ACEPTA la respuesta como CORRECTA."
    )


    prompt_template = (
        "TEXTO:\n{text}\n\n"
        "PREGUNTA:\n{question}\n\n"
        "RESPUESTA:\n{answer}\n\n"
        "¿La respuesta es correcta según el texto? Justifica y cita el fragmento
relevante.\n"
        "Devuelve SOLO el JSON pedido."
    )

    return Agent(
        llm_config=llm_config,
        system_prompt=system_prompt,
        prompt_template=prompt_template,
        name="Answer Verification Agent",
        description="Verifica si la respuesta a una pregunta es correcta según el
texto de la página."
    )
```