



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Diseño e implementación de una base de datos vectorial para la búsqueda semántica en manuales técnicos

Camilo Bilbao

camilo.bilbao@sansano.usm.cl

Catherine Gómez
Profesora Guía

David Larrondo
Profesor Correferente

Resumen: El acceso oportuno a información técnica en entornos industriales exige superar las limitaciones de la búsqueda léxica tradicional. El presente trabajo se centra en el Diseño e Implementación de una Base de Datos Vectorial para la búsqueda semántica en manuales técnicos. La solución desarrolla un robusto pipeline de ingeniería de datos que incluye una estrategia híbrida de extracción (texto digital y OCR mejorado), chunking optimizado y la indexación de representaciones vectoriales densas en FAISS. Se implementó una arquitectura de recuperación híbrida (Vectorial + BM25). El desempeño técnico fue sobresaliente, alcanzando una latencia promedio inferior a 25ms y validando la eficiencia de la arquitectura de la base de datos. No obstante, la validación de precisión operativa reveló que la recuperación pura es insuficiente, logrando un HitRate@5 de 43.3%. Esta caracterización empírica demuestra que, si bien la base de datos vectorial es una solución de ingeniería de alto rendimiento para la indexación y consulta rápida, la precisión semántica para la toma de decisiones crítica. Se proyecta evaluar si una capa generativa mejora la utilidad de lo recuperado, que aporte razonamiento contextual y desambiguación.

Palabras Clave: Búsqueda semántica, Embeddings, NLP, Inteligencia Artificial, Base de datos vectorial.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Diseño e implementación de una base de datos vectorial para la búsqueda semántica en manuales técnicos

Nombre del candidato(a): Camilo Bilbao Castro

Carrera / Grado: Ingeniería en informática

Campus: Viña del mar

Departamento: Electrotécnica e informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Catherine Gómez Barrera, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 16/10/2026

Firma:

Estudiante o Candidato(a):

Fecha: 16/10/2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

1 Introducción

1.1 Contexto y antecedentes

En el ámbito industrial, los manuales técnicos son esenciales para la operación y el mantenimiento de maquinaria, ya que contienen información detallada sobre procedimientos, seguridad y especificaciones técnicas. Sin embargo, estos documentos suelen ser extensos, complejos y dispersos en distintos formatos, lo que dificulta su consulta rápida por parte de técnicos y operarios. Esta situación puede retrasar la resolución de incidentes, incrementar el tiempo de inactividad de la maquinaria y generar pérdidas económicas significativas para la empresa.

Actualmente, muchas organizaciones dependen de procedimientos tradicionales para consultar estos manuales, como archivos PDF o impresos, lo que limita la eficiencia y la disponibilidad de la información. Aunque en la literatura se han explorado soluciones basadas en modelos de lenguaje y técnicas de búsqueda semántica, la implementación de sistemas que optimicen la organización y consulta de información técnica sigue siendo un desafío. Esto evidencia la necesidad de herramientas que permitan acceder de manera rápida, confiable y estructurada a la información contenida en manuales técnicos.

1.2 Definición del problema.

La problemática central identificada es la dificultad de acceder y consultar información técnica de manera eficiente. Manuales extensos y dispersos generan barreras para que técnicos y operarios encuentren información relevante de forma inmediata. Esta limitación retrasa la resolución de incidentes, incrementa los tiempos de detención de la maquinaria y eleva los costos operacionales.

Los métodos tradicionales para consultar documentos técnicos en formato PDF suelen basarse en la revisión manual del índice y la exploración de secciones potencialmente relevantes, con el fin de localizar información útil o una posible solución a un problema específico. Este proceso se ve particularmente limitado cuando los documentos se encuentran en formato físico o corresponden a archivos escaneados, debido a la imposibilidad de realizar búsquedas automatizadas.

En el caso de documentos PDF nativos, una práctica común consiste en utilizar búsquedas por coincidencia de palabras clave, mediante herramientas integradas del lector de PDF. Si bien este enfoque permite una localización rápida de términos específicos, presenta limitaciones significativas en relación con el contexto, ya que la presencia de una palabra no garantiza su relevancia semántica. En consecuencia, el usuario debe realizar una interpretación manual para determinar si el resultado encontrado corresponde efectivamente a la información requerida, pudiendo incluso tratarse del mismo término empleado con un significado distinto.

En este contexto, resulta necesario desarrollar un sistema que permita gestionar y organizar los manuales técnicos para facilitar la búsqueda de información de manera rápida y confiable. La solución debe optimizar el acceso a contenidos relevantes, reducir la dependencia de la experiencia individual y mejorar la eficiencia operativa en la gestión de la información técnica.

1.3 Breve descripción sobre la propuesta de solución

La propuesta de la tesina consiste en el diseño e implementación de una base de datos vectorial que permita realizar búsquedas semánticas en manuales técnicos. Esta base de datos transforma los contenidos de los manuales en vectores que pueden ser consultados de forma eficiente, facilitando la localización de información relevante incluso en documentos extensos.

El rol de ingeniería de datos incluye la extracción, limpieza y normalización de los documentos, la vectorización del contenido, la indexación en la base de datos y la optimización de las consultas para garantizar resultados rápidos y confiables.

Los límites del proyecto incluyen únicamente la preparación de los manuales, la construcción y gestión de la base de datos vectorial, y la implementación de la búsqueda semántica. Se excluyen aspectos como integración con sistemas de control industrial, desarrollo de interfaces de usuario avanzadas, predicción automática de fallas o implementación de modelos de lenguaje para generación de respuestas.

1.4 Objetivos Generales y Específicos de la Tesina

Objetivo General. Diseñar e implementar una base de datos vectorial para la búsqueda semántica en manuales técnicos, que permita organizar, indexar y consultar de manera eficiente la información contenida en dichos documentos, con el fin de mejorar la accesibilidad y reducir la latencia de recuperación de información relevante en el contexto industrial.

Objetivos Específicos

- Extraer y procesar manuales técnicos en distintos formatos, asegurando su limpieza y normalización para su posterior uso.
- Implementar un proceso de vectorización del contenido textual, utilizando representaciones numéricas que faciliten la búsqueda semántica.
- Diseñar la base de datos vectorial, considerando criterios de escalabilidad y rendimiento.
- Caracterizar el desempeño de la recuperación vectorial, vectorial con consultas enriquecidas e híbrida en un conjunto de consultas representativas.
- Documentar el diseño, la implementación y los resultados obtenidos, destacando su aplicabilidad en entornos industriales.

1.5 Justificación de proyecto

Relevancia técnica. El acceso oportuno a información técnica es un factor crítico en entornos industriales, donde el tiempo de inactividad de maquinaria puede traducirse en pérdidas económicas significativas. La implementación de una base de datos vectorial permite realizar búsquedas semánticas más eficientes que las consultas tradicionales basadas en coincidencia de palabras clave. Desde el punto de vista técnico, este proyecto contribuye al área de ingeniería de datos al aplicar técnicas modernas de representación y organización de información no estructurada, mejorando métricas como el tiempo de respuesta y la precisión en los resultados de búsqueda.

Beneficios. La propuesta facilita el acceso inmediato a información contenida en manuales técnicos, reduciendo la dependencia de la experiencia individual de los trabajadores y acelerando los procesos de diagnóstico y mantenimiento. Asimismo, ofrece una solución escalable y adaptable a diferentes volúmenes de información, que puede ser extendida a otros contextos donde la consulta de documentos técnicos sea crítica.

Innovación. El proyecto introduce un enfoque innovador al aplicar vectorización y bases de datos vectoriales específicamente al dominio de manuales industriales, un ámbito donde aún predominan soluciones tradicionales como archivos PDF o buscadores por palabras clave. Esta aproximación representa un avance hacia la búsqueda inteligente de información técnica, alineándose con las tendencias de transformación digital en la industria.

1.6 Metodología

Para el desarrollo de la tesina se adoptó una metodología ágil, inspirada en prácticas de Scrum, que permitió dividir el trabajo en iteraciones cortas y controladas, facilitando la retroalimentación y la mejora continua del sistema.

Las principales etapas fueron:

Sprint 1:

- Recolección y análisis de manuales técnicos en distintos formatos.
- Procesamiento y limpieza de datos, normalizando la información para asegurar consistencia.
- Vectorización del contenido, mediante el uso de representaciones numéricas del texto.

Sprint 2:

- Diseño e implementación de la base de datos vectorial, asegurando escalabilidad y eficiencia en las consultas.
- Integración de metadatos en la base de datos para entregar referencias del pdf al modelo

Sprint 3:

- Sistema de monitorización con parámetros detallados de la base de datos vectorial
- Configuración y ajuste de parámetros para mejorar la precisión y seguridad de las respuestas.
- Pruebas de rendimiento y validación de outputs generados por el modelo.
- Revisión de la estructura de embeddings y depuración de la base de conocimiento técnica.

Esta metodología se eligió porque facilita la iteración rápida sobre cada etapa del desarrollo, adaptándose a los resultados obtenidos y a las necesidades del sistema.

2 Marco Teórico.

2.1 Fundamentos la Ingeniería de Datos aplicada a la búsqueda semántica

La ingeniería de datos en este proyecto se enfoca en la construcción de un pipeline capaz de transformar manuales técnicos en representaciones vectoriales consultables. El proceso contempla la extracción de información desde documentos en distintos formatos, su limpieza y normalización, la digitalización de manuales escaneados mediante OCR.

En sistemas modernos de recuperación semántica, los documentos son fragmentados en bloques textuales manejables, los cuales son transformados en representaciones vectoriales densas mediante modelos de embeddings, para posteriormente ser indexados en estructuras optimizadas para consultas de similitud. Este enfoque se enmarca dentro de la recuperación de información neural (Mittra & Craswell, 2018) y ha sido ampliamente validado mediante modelos de recuperación densa basados en pasajes (Karpukhin, y otros, 2020). La generación de embeddings a nivel de oraciones o fragmentos se apoya en arquitecturas tipo Sentence-BERT (Reimers & Gurevych, 2019).

Este flujo permite convertir información no estructurada en datos organizados y accesibles, habilitando la búsqueda semántica como una alternativa más amigable a la búsqueda por palabras clave tradicionales.

2.2 Arquitectura de software

Capas y responsabilidades. El sistema utiliza una arquitectura modular en capas, donde cada componente cumple una función específica dentro del proceso de indexación semántica de manuales técnicos. Aunque el proyecto general se basa en Django, el módulo de vectorización opera de forma desacoplada, tomando como entrada los archivos PDF almacenados por la aplicación y generando estructuras de búsqueda optimizadas.

- Capa de acceso a datos:

Esta capa identifica los manuales disponibles a partir de los archivos almacenados en el directorio configurado por Django.

Se encarga de verificar la existencia de los archivos PDF y mantener un registro sincronizado de los documentos procesados mediante estructuras JSON auxiliares.

- Capa de procesamiento:

Encargada de la extracción de texto y tablas desde los archivos PDF, todas las herramientas utilizadas corresponden a librerías de python.

Aplica un enfoque híbrido de extracción, utilizando pdfplumber para contenido digital y pytesseract (OCR) para páginas escaneadas.

Antes del OCR, se mejora la calidad de las imágenes con OpenCV (ecualización, desenfoque, umbral adaptativo, corrección de inclinación).

- Capa de vectorización e indexación:

Genera los embeddings del texto procesado mediante modelos de Hugging Face y los almacena en una base de datos vectorial FAISS, dentro del directorio especificado por Django.

Esta capa también gestiona la actualización del índice, eliminando o añadiendo documentos según los cambios detectados en los archivos fuente.

Esta arquitectura modular favorece la mantenibilidad, ya que cada capa puede modificarse sin afectar al resto.

Además, su desacoplamiento respecto a Django permite ejecutar el proceso de vectorización de manera independiente (por ejemplo, como tarea en segundo plano o proceso batch), manteniendo sincronizada la información mediante los archivos de índice locales.

2.3 Tecnologías y frameworks utilizados

El desarrollo del sistema de procesamiento y vectorización de manuales técnicos se implementó íntegramente en Python, debido a su amplia disponibilidad de librerías en el ámbito de la inteligencia artificial, procesamiento de texto y visión por computador.

El sistema se estructura en torno a un pipeline modular que automatiza la lectura, extracción, vectorización y almacenamiento semántico de documentos técnicos.

A continuación, se describen las principales tecnologías, frameworks y herramientas empleadas en cada etapa del proceso.

Procesamiento de manuales técnicos. Los manuales técnicos representan la principal fuente de datos de este trabajo. Estos pueden encontrarse en formato PDF nativo o como escaneos en imagen. Para cubrir ambos escenarios, el pipeline implementa dos estrategias:

- PDFs nativos: se emplea pdfplumber para extraer el texto incrustado en el documento.
- Escaneos en imagen: se utilizan librerías como fitz (PyMuPDF) y Pillow para convertir cada página en imagen y aplicar normalizaciones básicas.

Previo al OCR, se realizan tareas de preprocesamiento como corrección de inclinación con deskew y skimage, y filtrados con OpenCV y NumPy. Estas transformaciones mejoran la precisión del OCR al reducir el ruido y facilitar la identificación de caracteres.

Finalmente, la digitalización del texto se realiza con pytesseract, que convierte las imágenes procesadas en texto plano.

Representación semántica y vectorización. Para otorgar utilidad a los datos textuales obtenidos y considerando su posterior integración al sistema, el texto extraído es sometido a un proceso de fragmentación mediante el uso de RecursiveCharacterTextSplitter. Este componente divide los documentos en segmentos de tamaño adecuado para su vectorización, lo cual resulta fundamental para el manejo de manuales extensos y para asegurar que los embeddings generados preserven el contexto semántico relevante.

Para la generación de embeddings se analizaron distintas estrategias basadas en modelos de arquitectura Transformer, tanto de ejecución local como mediante servicios en la nube. Con el objetivo de seleccionar la alternativa más adecuada para el prototipo, se elaboró una tabla comparativa (ver Tabla 1 *Comparativa de librería de embeddings*) que considera las ventajas y desventajas de cada enfoque.

Tabla 1 Comparativa de librería de embeddings

Tecnología Librería	Tipo	Ventajas	Desventajas	Decisión
HuggingFaceEmbeddings	Open source	Integración directa en Python/LangChain, buena calidad semántica, gratuito	Requiere recursos locales para embeddings grandes	Elegida
OpenAI embeddings	Cloud, API	Alta calidad semántica, mantenimiento mínimo	Pago por uso, latencia API	No seleccionada para prototipo local
SentenceTransformers	Open source	Amplia variedad de modelos preentrenados, buena performance local	Necesita GPU para embeddings grandes	Podría ser alternativa futura

Nota. Tabla de elaboración propia basada en la documentación de Hugging Face (Hugging Face, 2016), Open AI (OpenAI, 2015) y el trabajo de implementación de modelo Sentence-BERT (Reimers & Gurevych, 2019)

En el prototipo desarrollado se utiliza HuggingFaceEmbeddings como capa de integración provista por LangChain. Esta capa emplea internamente modelos Transformer preentrenados disponibles en Hugging Face, muchos de los cuales están implementados mediante la librería SentenceTransformers, el modelo específico que se va a utilizar es paraphrase-multilingual-MiniLM-L12-v2. La decisión de utilizar este modelo se fundamenta principalmente en limitaciones de hardware y consideraciones económicas, ya que permite ejecutar el proceso de vectorización de forma local sin incurrir en costos asociados a servicios externos.

Bases de datos vectoriales. Antes de evaluar las herramientas, se definieron tres directrices técnicas restrictivas basadas en la arquitectura del sistema:

- **Adecuación al prototipo (Ejecución Local):** Se priorizaron soluciones de código abierto que permitieran ejecución local para evitar la dependencia de servicios de terceros y costos asociados a APIs de pago.
- **Complejidad de Configuración (Overhead):** Dado que el proyecto es un prototipo inicial, se debían descartar herramientas de distribución nativa que introdujeran una complejidad de configuración excesiva o un overhead innecesario para el volumen de datos actual.
- **Ruta de Escalabilidad:** Aunque el corpus actual es acotado (aproximadamente 4.500 vectores), la herramienta elegida debía soportar algorítmicamente la transición futura desde índices de búsqueda exhaustiva hacia índices aproximados y agrupados (como Inverted File Index) para entornos masivos.

Considerando estas directrices y basándose en la documentación oficial de FAISS, Milvus, Pinecone y Weaviate, se elaboró la siguiente comparativa

Tabla 2 Comparativa de bases de datos vectoriales

Tecnología	Tipo	Ventajas	Desventajas	Decisión
FAISS	Local / open source	Mayor eficiencia en búsquedas ANN, escalable, integración sencilla con Python	No distribuido de forma nativa, requiere manejo manual de persistencia	Elegida
Milvus	Distribuida, open source	Distribución nativa, escalable horizontalmente, soporte cloud	Mayor complejidad de configuración, overhead en prototipos pequeños	No seleccionada
Pinecone	Vector DB as a service	Fácil integración cloud, mantenimiento mínimo	Servicio de pago, dependiente de terceros	No seleccionada
Weaviate	Distribuida, open source	Incluye NLP integrado, búsqueda semántica avanzada	Configuración compleja, menos control sobre índices	No seleccionada

Nota. Tabla de elaboración propia basada en la documentación oficial de FAISS (Meta AI, 2017), Milvus (Milvus, 2017), Pinecone (Pinecone Systems, Inc, 2019) y Weaviate (Weaviate, 2019).

FAISS fue seleccionada por cumplir adecuadamente con los requerimientos de búsqueda semántica del prototipo, ofreciendo alto rendimiento en entornos locales y reducida complejidad de configuración, lo que resulta coherente con la escala del sistema desarrollado.

La integración con LangChain permite realizar tanto la indexación como la consulta de los vectores de manera sencilla y modular, adaptándose a los requerimientos de un sistema de búsqueda semántica. El análisis detallado sobre los algoritmos de indexación de FAISS, sus límites de escalabilidad matemática y las estructuras de optimización implementadas en el sistema se abordan en profundidad en la sección 3.1 (Diseño de Componentes)

Orquestación con LangChain. El framework LangChain se utiliza como capa de orquestación dentro del pipeline. A través de sus componentes, se gestiona la carga y estructuración de los documentos, la fragmentación del texto, la generación de embeddings mediante HuggingFaceEmbeddings y su indexación en la base de datos vectorial FAISS.

Aunque LangChain es conocido principalmente por su integración con modelos de lenguaje de gran tamaño (LLMs), en este proyecto se emplea específicamente como herramienta de procesamiento de datos semánticos, actuando como conector y facilitador de las distintas etapas del pipeline.

La presente etapa se centra en la preparación de los manuales técnicos, la vectorización y la estructuración de la base de datos, dejando para etapas posteriores la integración con modelos de lenguaje y técnicas de recuperación semántica.

Búsqueda Léxica (Sparse Retrieval). El algoritmo BM25 (Best Match 25) es un modelo de ranking léxico que se considera el estándar de facto en sistemas de Recuperación de Información (IR) basados en la frecuencia y la rareza de las palabras. Funciona como una evolución avanzada del modelo TF-IDF (Frecuencia de Término–Frecuencia Inversa de Documento). Su mecanismo proporciona un puntaje de ranking sin cota de relevancia. En el contexto de la búsqueda híbrida, BM25 cumple una función esencial: actúa como un componente de seguridad y precisión literal, garantizando la recuperación de coincidencias exactas de códigos, modelos, y terminología técnica que los modelos de embeddings semánticos a menudo fallan en diferenciar debido al fenómeno de la dilución (Robertson, Jones, Zaragoza, & Walker, 2009).

2.4 Estado del arte

En la última década, la Inteligencia Artificial (IA) y el Procesamiento de Lenguaje Natural (NLP) han abierto nuevas posibilidades para abordar esta problemática. La búsqueda semántica ha emergido como técnica fundamental, permitiendo comprender el significado de las consultas en lugar de coincidir únicamente palabras exactas. Los sistemas de recuperación semántica que emplean representaciones vectoriales permiten capturar relaciones conceptuales entre textos, facilitando la recuperación de información incluso en ausencia de coincidencia literal de términos (Reimers & Gurevych, 2019).

Diversos enfoques recientes incluyen la utilización de modelos de lenguaje preentrenados, como los de HuggingFace o OpenAI, que generan embeddings de alta calidad para documentos técnicos. Herramientas como FAISS, Milvus o Weaviate han sido empleadas para almacenar y consultar estos vectores, optimizando la búsqueda de información en colecciones extensas. Sin embargo, entrenar un modelo propio desde cero para cada colección de manuales requiere gran cantidad de datos, recursos computacionales y tiempo, además de mantenimiento constante para adaptarse a nuevos documentos o dominios específicos.

Por esta razón, el presente proyecto adopta un enfoque basado en modelos preentrenados y pipelines modulares, combinando OCR avanzado, embeddings de HuggingFace y FAISS. Este enfoque permite acelerar la implementación, reducir costos y mantener alta calidad semántica sin necesidad de entrenar modelos desde cero. Además, su diseño desacoplado facilita la actualización y expansión del sistema, garantizando escalabilidad y eficiencia en la recuperación de información técnica.

En trabajos recientes como HiQA se ha demostrado que los sistemas basados en búsquedas semánticas permiten responder sobre múltiples documentos, mejorando la precisión respecto a métodos basados sólo en coincidencia de palabras. Sin embargo, estos sistemas también enfrentan limitaciones cuando hay documentos muy similares, la recuperación puede devolver fragmentos con significado semántico similar pero no necesariamente relevantes, lo que complica garantizar la exactitud de las respuestas (Chen, Gao, Song, & Tan, 2024).

Uno de los problemas críticos en sistemas de recuperación semántica basados en embeddings es la fragmentación del texto (chunking). La selección del tamaño y forma de los fragmentos es determinante, si los “chunks” son muy grandes, pueden confluír múltiples conceptos en un solo vector, diluyendo la especificidad semántica; por el contrario, si son muy pequeños, se corre el riesgo de perder contexto relevante. Esta pérdida de coherencia contextual puede degradar la relevancia de los resultados. En contextos técnicos, además, vocablos o matices específicos del dominio pueden no

quedar bien representados por embeddings genéricos, lo que reduce precisión y puede producir respuestas incorrectas o incompletas. Adicionalmente, el enfoque basado exclusivamente en embeddings + búsqueda vectorial (sin complementarlo con métodos léxicos, re-ranking, metadatos o lógica de estructura documental) puede fallar cuando se requiere recuperación exacta, referencias precisas o preservación de relaciones estructurales (fechas, versiones, jerarquías, definiciones técnicas, etc.). En esos casos, la similitud semántica no garantiza relevancia o exactitud, y la solución ideal pasa por arquitecturas híbridas o complementarias que integren múltiples tipos de indexación o validación (Gomez-Cabello, y otros, 2025).

Más allá de los enfoques vectoriales, la investigación en recuperación de información ha explorado paradigmas alternativos basados en grafos de conocimiento. En estos sistemas, el conocimiento se estructura como una red de nodos (entidades, conceptos) y aristas (relaciones explícitas como "es-parte-de", "requiere", "es-compatible-con"), permitiendo búsquedas mediante recorrido de grafos (graph traversal) o razonamiento multisalto que siguen caminos lógicos entre conceptos (Färber, Bartscherer, Menne, & Rettinger, 2018).

Para dominios técnicos como manuales, este enfoque ofrece ventajas teóricas: preserva relaciones jerárquicas y dependenciales que son cruciales, y permite responder consultas complejas que requieren encadenar varios hechos (por ejemplo, "¿Qué componentes deben desmontarse antes de acceder al sensor X?"). Sin embargo, su implementación práctica enfrenta barreras significativas. La construcción del grafo requiere un proceso costoso de extracción de entidades y relaciones (entity and relationship extraction), que típicamente depende de modelos de lenguaje grandes con alto consumo computacional, o de pipelines de NLP específicas por dominio (Hogan, y otros, 2021). Además, las consultas por traversal pueden tener complejidad computacional mayor que la búsqueda aproximada por vecinos más cercanos (ANN) en índices vectoriales optimizados, especialmente a escala. Finalmente, la actualización dinámica del grafo con nuevos documentos es más compleja que la re-indexación vectorial.

Estas limitaciones, sumadas al objetivo principal del presente trabajo que es "diseñar un sistema balanceado entre precisión semántica, eficiencia y facilidad de implementación" justifican la elección de la búsqueda semántica basada en embeddings e índices vectoriales como núcleo de la solución. No obstante, se reconoce el valor de los grafos para capturar relaciones estructurales, y se sugiere que futuras extensiones podrían explorar arquitecturas híbridas que combinen la recuperación semántica inicial vía vectores con un refinamiento posterior que explote relaciones gráficas predefinidas para consultas específicas (Xie, Huang, Du, Zhang, & Huang, 2022).

3 Diseño e Implementación

Esta sección describe el diseño e implementación del sistema de búsqueda semántica en manuales técnicos, enfocándose en el procesamiento, indexación y recuperación de información basada en embeddings.

3.1 Diseño de Componentes

Arquitectura y flujo de recuperación semántica. El sistema implementa una arquitectura modular basada en tres componentes principales que conforman el pipeline de recuperación semántica. Esta separación permite flexibilidad, mantenibilidad y actualización independiente de cada componente.

Componente 1: Procesamiento de Documentos. El módulo de procesamiento constituye la etapa inicial del pipeline, encargada de la ingesta y transformación de manuales técnicos (formato PDF) en estructuras de datos normalizadas. Este componente implementa una estrategia híbrida de extracción que se adapta dinámicamente al tipo de contenido:

- **Extracción de Texto y OCR:** Para documentos digitales nativos, se utiliza la librería `pdfplumber`, permitiendo una extracción directa que preserva la jerarquía lógica (encabezados, párrafos y listas). En el caso de documentos escaneados o imágenes, el sistema activa automáticamente un flujo de reconocimiento óptico de caracteres (OCR) mediante Tesseract. Este subproceso incluye una fase de pre-procesamiento de imagen con OpenCV (corrección de sesgo o skew, normalización de contraste) y aplica múltiples configuraciones de segmentación de página (PSM) para maximizar la tasa de reconocimiento. Además, se implementa un mecanismo de fallback automático que deriva al OCR si la extracción de texto estándar falla o retorna resultados vacíos.
- **Estructuración de Elementos Complejos:** El sistema identifica y procesa elementos no textuales críticos. Las tablas se detectan y extraen preservando su estructura matricial mediante `pdfplumber.extract_tables()`. Paralelamente, se extraen metadatos clave (marca, modelo, categoría) y se segmenta el documento mediante expresiones regulares (regex) para etiquetar secciones prioritarias como "Seguridad", "Mantenimiento" o "Especificaciones Técnicas".

El resultado de esta etapa es una lista de objetos `Document` enriquecidos, cuya estructura de metadatos incluye la fuente, número de página, tipo de contenido (textual/scanned), e indicadores booleanos de relevancia (`is_important_section`), facilitando el filtrado posterior (ver Figura 1).

```
{
  "page_content": "Contenido textual extraído del manual...",
  "metadata": {
    "source": "Manual Operación.pdf",
    "page": 12,
    "type": "textual",
    "section": "Mantenimiento",
    "manual_id": "MN-001",
    "brand": "Caterpillar",
    "equipment": "Excavadora"
  }
}
```

Figura 1. Esquema lógico del objeto Document en memoria

Componente 2: Chunking y Vectorización. Este componente es responsable de transformar el contenido textual extraído en representaciones vectoriales densas, aptas para la búsqueda semántica.

- Estrategia de Fragmentación (Chunking): Se emplea el algoritmo RecursiveCharacterTextSplitter con una estrategia jerárquica que prioriza los límites semánticos (encabezados y saltos de párrafo) sobre el conteo estricto de caracteres. La configuración base utiliza un tamaño de fragmento (CHUNK_SIZE) de 1000 caracteres con un solapamiento (CHUNK_OVERLAP) de 200 caracteres. Este solapamiento es crítico para asegurar que el contexto no se pierda en los cortes arbitrarios. Las tablas reciben un tratamiento diferenciado, preservándose como unidades indivisibles para mantener la coherencia de filas y columnas.
- Generación de Embeddings: Para la vectorización se seleccionó el modelo sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2. Este modelo Transformer genera vectores densos de 384 dimensiones, optimizados para baja latencia (~10ms por chunk en CPU) y con soporte multilingüe, lo que resulta esencial para procesar documentación técnica que combina español e inglés.

Componente 3: Base de Datos Vectorial. Almacenamiento e indexación de embeddings para búsqueda eficiente.

Funcionalidades:

1. Indexación con FAISS

El índice FAISS basado en IndexFlatL2 permite realizar una búsqueda exhaustiva de los vectores almacenados según la distancia euclidiana, evitando las aproximaciones propias de determinados índices de búsqueda vectorial. Sin embargo, esta propiedad no garantiza por sí misma la relevancia semántica de los resultados.

Como trabajo futuro para entornos de alta escalabilidad, se propone la migración hacia IndexIVF (Inverted File Index). Este índice utiliza una estructura de clústeres (celdas de Voronoi) para reducir el espacio de búsqueda, mejorando significativamente la velocidad. No obstante, su implementación implica un compromiso: al ser una búsqueda aproximada (ANN), existe el riesgo de pérdida de precisión si el vector más relevante se encuentra en el límite de un clúster no visitado, lo que requiere una configuración cuidadosa del parámetro de búsqueda (nprobe) y un proceso de entrenamiento previo del índice.

2. Almacenamiento Persistente

Para asegurar la persistencia de los datos, el sistema genera y almacena dos archivos principales en el directorio `data/faiss_index/`: un archivo `.faiss` que contiene la estructura del índice vectorial optimizado y un archivo `.pkl` (Pickle) que almacena los metadatos asociados a los fragmentos de texto. Adicionalmente, esta estructura soporta el versionado de índices, facilitando las labores de auditoría y pruebas de regresión.

3. Consultas de Similitud

El flujo de consulta se inicia con la vectorización de la interrogante del usuario, utilizando estrictamente el mismo modelo empleado durante la fase de indexación para garantizar la coherencia semántica. Posteriormente, se ejecuta una búsqueda de los k vecinos más cercanos (k -Nearest Neighbors) en el índice FAISS, recuperando los fragmentos (chunks) originales mediante sus metadatos asociados. Los resultados se ordenan por similitud de coseno normalizada, seleccionando típicamente los 5 fragmentos más relevantes ($k=5$). Adicionalmente, el sistema admite un umbral de corte (thresholding) opcional para descartar resultados que no cumplan con un nivel mínimo de similitud.

4. Monitoreo y Estadísticas

Para asegurar la observabilidad del sistema, se implementó un módulo de registro que almacena métricas clave en el archivo `vector_stats.json` para su análisis histórico. Las principales métricas recopiladas incluyen:

- Volumen de datos: Conteo total de fragmentos indexados (`total_chunks`) y cantidad de manuales únicos procesados (`unique_manuales`).
- Calidad de fragmentación: Estadísticas de distribución de tamaños de los chunks (mínimo, máximo, media y desviación estándar) para detectar anomalías en el corte.
- Integridad de metadatos: Cobertura porcentual de campos de metadatos (`metadata_coverage`) para validar la calidad de la información estructurada.
- Métricas de procesamiento: Desglose entre documentos textuales nativos y documentos escaneados procesados vía OCR (`processing_metrics`).

Patrones y buenas prácticas aplicadas. El desarrollo del sistema se fundamentó en principios de ingeniería de software orientados a la mantenibilidad y la robustez del procesamiento de datos. Se destacan tres pilares principales:

Modularidad. La arquitectura desacopla las responsabilidades en funciones especializadas dentro del módulo `vector_service.py` (Figura 2). Esta separación permite aislar la lógica de extracción, limpieza y vectorización, facilitando la ejecución de pruebas unitarias y permitiendo el reemplazo futuro de componentes (como el motor de OCR) sin afectar al orquestador principal.

```
vector_service.py
├── smart_pdf_extraction()      # Lectura + OCR + Extracción
├── get_optimized_text_splitter() # Configuración de chunking
├── prepare_vectorstore()      # Orquestador principal
└── VectorDBMonitor           # Monitoreo de estadísticas
```

Figura 2. Estructura modular del servicio vectorial

Reutilización y Consistencia. Para evitar la duplicación de código y garantizar la integridad de los datos, se centralizaron las funciones de procesamiento. Tanto los documentos nuevos como los históricos atraviesan estrictamente el mismo pipeline, asegurando que la estrategia de fragmentación (chunking) sea homogénea en toda la base de conocimiento.

Observabilidad. La clase `VectorDBMonitor` permite auditar el estado del índice en tiempo real.

A continuación, se detallan las métricas capturadas (Tabla 3) y un ejemplo de salida del sistema tras la indexación del corpus de validación (Figura 3).

Tabla 3. Métricas clase `VectorDBMonitor`

Métrica	Propósito	Uso
<code>chunk_sizes</code>	Distribución de tamaños	Validar configuración de chunking
<code>metadata_coverage</code>	% campos completos	Detectar datos incompletos
<code>processing_metrics</code>	OCR vs texto	Evaluar calidad de la fuente
<code>manuals_breakdown</code>	Chunks por manual	Identificar desbalances
<code>indexed_at</code>	Timestamp indexación	Auditoría y versioning

```
{
  "timestamp": "2025-11-29T03:52:32.656493",
  "total_chunks": 4352,
  "manuals_breakdown": {
    "LIEBHERR_Bulldozer_Oper_PR766.pdf": 589,
    "LIEBHERR_Cargadora_Mant_L538.pdf": 259,
    "LIEBHERR_Cargadora_Mant_L550.pdf": 2283,
    "LIEBHERR_Cargadora_Oper_L538.pdf": 620,
    "LIEBHERR_Excavadora_Oper_R934.pdf": 601
  },
  "chunk_sizes": {
    "mean": 744.1505055147059
  }
}
```

Figura 3. Ejemplo reporte de estadísticas para cinco manuales

Consideraciones técnicas. El diseño del sistema aborda desafíos críticos relacionados con la heterogeneidad de los documentos y la preservación del contexto estructural. A continuación, se detallan las decisiones de ingeniería adoptadas.

Estrategia de Fragmentación (Chunking). Durante la fase de experimentación, se identificó que el uso de fragmentos estándar (250-700 caracteres) segmentaba excesivamente las tablas técnicas, rompiendo la relación lógica entre los encabezados y las filas de datos. Para mitigar este problema, se estableció una configuración fija de `CHUNK_SIZE = 1000` caracteres con un solapamiento (`CHUNK_OVERLAP`) de 200 caracteres.

Esta decisión de diseño se fundamenta en tres criterios:

1. **Integridad Tabular:** El tamaño ampliado permite encapsular tablas completas o secciones significativas de las mismas dentro de un único vector, preservando el contexto necesario para la recuperación precisa.
2. **Límite de Tokens:** Un tamaño de 1000 caracteres equivale aproximadamente a 200-250 tokens, lo cual se mantiene dentro del límite de entrada (Max Sequence Length) del modelo MiniLM (512 tokens), evitando truncamientos de información.
3. **Redundancia de Contexto:** El solapamiento de 200 caracteres asegura que no se pierda información crítica en los cortes de página o secciones.

Soporte Multilingüe. La vectorización se fundamenta en el modelo paraphrase-multilingual-MiniLM-L12-v2. Esta elección permite procesar consultas y documentos indistintamente en español o inglés mediante un único espacio vectorial compartido. Si bien este enfoque simplifica la arquitectura, se reconoce como limitación el posible sesgo hacia idiomas predominantes en el entrenamiento del modelo.

Ciclo de Vida del Índice. Para garantizar la consistencia de los datos sin requerir costosas re-indexaciones completas, se implementó un flujo de actualización incremental basado en eventos:

1. **Ingesta (Nuevo Manual):** Se procesa el PDF, se generan los embeddings y se añaden al índice mediante `FAISS.add_documents()`, persistiendo el resultado inmediatamente.
2. **Modificación:** Se detectan cambios mediante hashing del archivo fuente. El sistema elimina los vectores obsoletos y re-procesa únicamente el documento afectado.
3. **Eliminación:** Se identifican los chunks asociados mediante metadatos y se purgan del índice.
4. **Optimización de Rendimiento.** Dado el costo computacional de los modelos Transformer, se aplicaron técnicas de ingeniería para reducir la latencia y mejorar la escalabilidad (ver Tabla 4).

Tabla 4. Estrategias de optimización implementadas

Optimización	Técnica	Impacto
Batch processing	Procesar múltiples PDFs paralelos	Reducción en tiempo de indexación.
Embedding cache	Reutilizar embeddings del modelo cargado	Reducción en overhead de carga.
L2 normalization	Convertir distancia L2 a similitud coseno	Mejora en la consistencia semántica del ranking.
Index persistence	Guardar índice en disco	Eliminación de tiempos de reentrenamiento al inicio.

Manejo de Errores y Recuperación. El sistema implementa un patrón de programación defensiva para asegurar la disponibilidad ante fallos en los datos de entrada, gestionando excepciones críticas sin detener el servicio.

Casos críticos:

1. PDF corrupto

```
try:
    docs = smart_pdf_extraction(pdf_path)
except Exception as e:
    logger.error(f"Error procesando {pdf_path}: {e}")
    return [] # Omisión controlada
```

Figura 4. Recuperación ante PDFs corruptos

2. Corrupción de Índice Vectorial

```
try:
    vectorstore = FAISS.load_local(FAISS_INDEX_PATH, ...)
except (FileNotFoundError, RuntimeError) as e:
    logger.warning(f"Error al cargar el índice: {e}. Se reconstruirá.")
    vectorstore = None # Fuerza la reconstrucción desde cero
```

Figura 5. Manejo de excepción índice vectorial corrupto

3. Fallo de OCR

```
if text and len(text) > 50:
    # Procesamiento de texto nativo
    docs.append(Document(page_content=full_text, ...))
else:
    # Activación de flujo de OCR ante ausencia de texto
    logger.info(f"Usando OCR en la página {page_number + 1}...")
    ocr_text = extract_with_ocr_enhanced(fitz_page, page_number + 1)
```

Figura 6. Manejo de fallo en OCR

3.2 Detalles del desarrollo de la búsqueda semántica

Esta sección profundiza en la operativización del diseño, detallando el flujo de datos, las estrategias de mantenimiento del índice y los criterios de validación.

Pipeline Automatizado. El sistema implementa un flujo de procesamiento secuencial (pipeline) dividido en seis fases críticas, diseñado para transformar documentos no estructurados en activos de información recuperables (Figura 7).

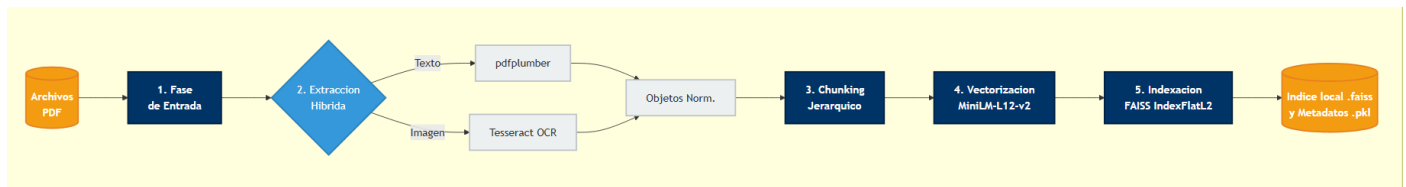


Figura 7. Creación del Índice - Pipeline Automatizado

- Fase de Entrada: El proceso se inicia con la detección de archivos PDF almacenados en la base de datos de gestión, actuando como disparador del flujo.
- Fase de Procesamiento: Se ejecuta la extracción híbrida mediante `smart_pdf_extraction()`, discriminando dinámicamente entre texto digital (vía `pdfplumber`) y contenido escaneado (vía `Tesseract OCR`). El resultado es una lista de objetos normalizados enriquecidos con metadatos descriptivos.
- Fase de Fragmentación (Chunking): Los documentos se segmentan utilizando una estrategia jerárquica que preserva encabezados.
- Fase de Vectorización: Cada fragmento validado es transformado en un vector denso de 384 dimensiones utilizando el modelo `MiniLM-L12-v2`.
- Fase de Indexación: Los vectores resultantes se persisten en disco mediante la estructura `IndexFlatL2` de `FAISS`, generando simultáneamente el registro de metadatos asociado.

Fase de Consulta (Runtime): En tiempo de ejecución, las consultas del usuario son vectorizadas con el mismo modelo y comparadas mediante similitud de coseno para recuperar los $k=5$ fragmentos más relevantes (Figura 8).

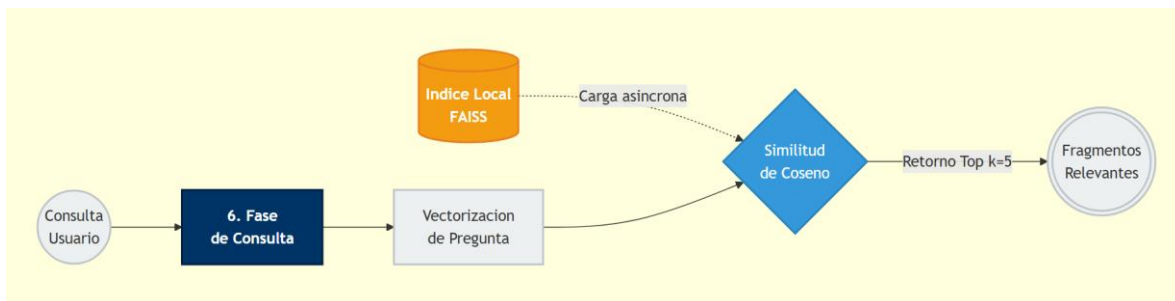


Figura 8. Consultas - Tiempo de Ejecución

Versionado de Modelos y Embeddings. Un desafío crítico en sistemas vectoriales es la incompatibilidad entre índices generados por distintas versiones de modelos de embeddings. Para mitigar este riesgo, se implementó un mecanismo de validación de metadatos en la cabecera del índice (Figura 9).

Problema: Cambios en modelos crean incompatibilidad con índices existentes.

```
{
  "embedding_model": "sentence-transformers/paraphrase-multilingual-
MiniLM-L12-v2",
  "model_version": "0.4.1",
  "embedding_dimension": 384,
  "indexed_date": "2025-12-11T10:30:00Z"
}
```

Figura 9. Estructura de metadatos para versionado.

El sistema verifica en tiempo de carga (load_time) que la firma del modelo actual coincida con la del índice almacenado. Si se detecta una discrepancia (ej. actualización de la librería sentence-transformers), el sistema emite una alerta crítica y sugiere la reconstrucción del índice, evitando así la degradación silenciosa de la calidad de búsqueda.

Definición de Métricas y Validación. Para evaluar la efectividad del sistema de recuperación, se seleccionó la métrica HitRate@k (Tasa de Acierto en el corte \$k\$). A diferencia de métricas orientadas al ranking estricto como NDCG (Normalized Discounted Cumulative Gain), el HitRate prioriza la capacidad de recuperación (Recall), lo cual es prioritario en un contexto de mantenimiento donde lo crucial es encontrar la información, independientemente de si aparece en la primera o tercera posición:

$$HR@5 = \frac{1}{|Q|} \sum_{q \in Q} \Pi(hit(q, k) > 0)$$

Donde:

- Q es el conjunto de resultados de prueba
- Π es una función que indica valor 1 si se encuentra un resultado relevante en el top k y 0 en caso contrario

Se estableció un umbral de aceptación de $HR@5 \geq 0.7$, lo que implica que el sistema debe encontrar al menos un fragmento relevante en las primeras 5 opciones para el 70% de las consultas.

La definición de este umbral crítico no es arbitraria y responde directamente a las necesidades del entorno industrial descritas en la formulación del problema:

Primero, el límite de evaluación de cinco opciones ($k=5$) se establece para minimizar la carga cognitiva del operario. En un escenario de maquinaria detenida, forzar al usuario a discriminar entre más de cinco fragmentos de texto resultaría en una ineficiencia similar a la de los procedimientos tradicionales de búsqueda.

Segundo, el porcentaje del 70% (0.7) representa el límite mínimo de viabilidad operativa. Un sistema de soporte a la toma de decisiones técnicas que falle en recuperar la información relevante en más del 30% de las consultas corre el riesgo de generar rechazo tecnológico por parte del usuario final, perdiendo confiabilidad operativa. Por lo

tanto, alcanzar este umbral es el requisito indispensable para considerar que el motor de búsqueda vectorial es funcional de manera autónoma.

Complementariamente, la efectividad de recuperación ($HR@5$) se formaliza estadísticamente como una variable aleatoria dicotómica de Bernoulli $Y_i \in \{0, 1\}$ definida para cada consulta $i \in \{1, 2, \dots, N\}$ sobre un conjunto de prueba de $N = 30$ evaluaciones, donde $Y_i = 1$ (Acierto o Hit) y $Y_i = 0$ (Fallo o Miss). La estimación puntual global de la tasa de acierto ($\hat{p}$) se expresa como:

$$\hat{p} = HR@5 = \frac{1}{N} \sum_{i=1}^N Y_i$$

A partir de esta formalización, se cuantifica la latencia de respuesta (L) en milisegundos (ms) y se caracterizan los parámetros de dispersión muestral mediante intervalos de confianza binomiales.

Monitoreo y estadísticas en tiempo de ejecución. Para asegurar la salud operativa del sistema y la calidad del servicio, se definieron puntos de control (Health Checks) y un esquema de métricas en tiempo real. A continuación, se presentan las estadísticas basales del índice y el desempeño observado durante la fase de validación.

Estado Inicial del Índice. Las siguientes métricas corresponden a la consolidación del corpus de validación (Liebherr):

- Manuales indexados: 5 documentos.
- Total de chunks: 4,352 fragmentos vectorizados.
- Tamaño promedio de chunk: 744 caracteres. (Este valor confirma la adherencia a la estrategia de preservación de tablas definida en el diseño).
- Cobertura de metadatos: 100% en campos críticos (Marca, Modelo, Categoría).

Métricas de Consulta (Desempeño en Runtime). El sistema registra la telemetría de cada interacción en el archivo `vector_stats.json` para monitorear la eficiencia del motor en tiempo real. Durante las trazas de prueba en tiempo de ejecución, se observaron los siguientes indicadores de desempeño:

- Latencia Promedio: 18 ms. El uso de filtros por metadatos (brand + model) permite tiempos de respuesta por debajo de los 20 ms, cumpliendo holgadamente con el SLA objetivo del sistema ($< 25 ms$).
- Tasa de Recuperación (HitRate Global): El seguimiento de telemetría permite registrar el comportamiento del indicador `HitRate@5`, identificando tempranamente la degradación de precisión ocasionada por la brecha lingüística en documentación en inglés, aspecto que se analiza en profundidad en la evaluación experimental de tres fases (Sección 3.3).

Ejemplo de Traza de Ejecución: Para la consulta ID 3 ("¿Qué significa el testigo de advertencia de temperatura de aceite hidráulico?"), el monitor registró:

- Tiempo de respuesta: 19ms.
- Estado: HIT (Recuperación exitosa).
- Score de relevancia: 15.34 (Distancia L2).
- Fuente recuperada: LIEBHERR_Buldocer_Oper_PR766.pdf (Pág. 89).
- Observación: El sistema logró asociar semánticamente el término "testigo" con "Símbolo de presión", superando inconsistencias de formato en el texto origen.

Alertas Preventivas. Finalmente, se configuraron reglas de negocio para detectar anomalías en el servicio productivo:

- Degradación de relevancia: Alerta si la distancia promedio del primer resultado (Rank 1) supera el umbral de 20.0 (indicando baja similitud semántica).
- Latencia excesiva: Alerta si el tiempo de vectorización + búsqueda supera los 500ms, lo cual indicaría la necesidad de particionar el índice o escalar horizontalmente.
- Exceso de OCR: Alerta si más del 30% de los resultados provienen de fuentes escaneadas, sugiriendo una revisión de la calidad de los documentos fuente.

3.3 Estrategias de evaluación y validación

La validación del sistema se llevó a cabo en tres fases iterativas, permitiendo aislar variables y medir el impacto de distintas estrategias de recuperación: (1) Búsqueda Vectorial con consultas simples, (2) Búsqueda Vectorial con expansión de consultas y (3) Búsqueda Híbrida.

Las tres fases se evaluaron sobre el mismo corpus indexado (5 manuales, 4.352 fragmentos) y las mismas 30 consultas, bajo las mismas condiciones de recuperación: a cada consulta se le asoció el modelo y la categoría del manual esperado, usados como filtro sobre los resultados recuperados, de modo que el sistema opera sobre un manual conocido por el usuario. Se devuelven hasta $k=5$ fragmentos, pudiendo ser menos cuando el filtro descarta resultados; una consulta sin resultados tras el filtro se registró como 'NO'. La relevancia de cada fragmento fue juzgada manualmente revisando congruencias con el contenido de los manuales. Lo que varía entre fases es la formulación de la consulta y el método de recuperación, como se detalla a continuación.

Fase 1: Línea Base (Búsqueda Vectorial Pura). En la primera iteración, se evaluó el desempeño del motor vectorial utilizando consultas "crudas" (lenguaje natural simple), simulando la entrada de un operario sin conocimientos técnicos profundos.

Resumen Cuantitativo (Línea Base). Tras una revisión manual de los resultados marcados preliminarmente como "Hit", se reclasificaron como "Falsos Positivos" aquellos fragmentos que, aunque temáticamente cercanos, no contenían la respuesta operativa específica (Tabla 5).

Tabla 5. Matriz de resultados consultas simples

Categoría de Prueba	Total	Hits Reales	Falsos Positivos	Miss	HitRate Real
Seguridad	9	2	3 (ID 1, 3, 5)	4	22.2%
Operación (Español)	10	5	2 (ID 12, 17)	3	50.0%
Mantenimiento (Inglés)	8	1	2 (ID 20, 27)	5	12.5%
Troubleshooting	3	0	1 (ID 30)	2	0.0%
GLOBAL	30	8	8	14	26.7%

Diagnóstico de Fallas (Línea Base). La auditoría de los "Falsos Positivos" (26.6% de los casos) permitió identificar las limitaciones cognitivas del modelo de embeddings cuando opera sin contexto suficiente:

1. Generalización Excesiva (Topic Drift):

- Caso ID 12: Ante la consulta "¿Qué significa el testigo luminoso...?", el sistema recuperó tablas de viscosidad de aceite. El modelo priorizó la coincidencia de los términos "aceite" y "temperatura", fallando en distinguir entre una propiedad física y una señal de advertencia.
- Caso ID 17: Para "Patrón de control ISO", se recuperaron secciones sobre el "mando de traslación" (pedales). El sistema asoció el campo semántico de "palancas", pero careció de especificidad.

2. Asociación de Riesgo vs. Mitigación:

- Caso ID 1: Ante la consulta sobre "Extintores", el sistema priorizó las causas del incendio ("Bomba de llenado") sobre las medidas de mitigación.

Fase 2: Enriquecimiento de Consultas. Se evaluó la hipótesis de que la brevedad de las consultas limitaba el contexto. Se utilizó un set de preguntas sometidas a Expansión de Consulta (Query Expansion) manual, añadiendo terminología técnica específica, un resumen de los resultados se encuentra en la Tabla 6.

Tabla 6. Matriz de resultados consultas enriquecidas

Categoría de Prueba	Total	Hits Reales	Falsos Positivos	Miss	HitRate Real
Seguridad Crítica	9	4	2 (ID 1, 3)	3	44.4%
Operación (Español)	10	6	1 (ID 12)	3	60.0%
Mantenimiento (Inglés)	8	1	0	7	12.5%*
Troubleshooting	3	0	1 (ID 30)	2	0.0%
GLOBAL	30	11	4	15	36.7%

***Nota:** El único acierto en Mantenimiento (ID 22) se logró porque la consulta contenía términos técnicos en inglés.

Análisis de Fenómenos Semánticos (Efectos Adversos). Si bien hubo una mejora (+10.1%), el enriquecimiento reveló comportamientos no lineales en el modelo de embeddings:

- Fenómeno de Dilución Semántica: Contrario a la intuición, añadir detalles descriptivos (ej. Query ID 1: "fuegos eléctricos", "agente extintor", "ubicación") incrementó el ruido. El vector resultante se "contaminó" con términos del contexto (eléctrico, fuego), provocando que el sistema recuperara el Sistema Eléctrico en lugar del Equipo de Extinción. Esto sugiere que, debido a la estrategia de Mean Pooling (promedio de vectores) utilizada para generar el embedding final, la representación semántica densa sufre de compresión de información, donde los términos de alta frecuencia del contexto diluyen la señal de los términos específicos.
- Confusión de Fluidos Técnicos: En la consulta ID 12 enriquecida ("Icono termómetro", "Aceite hidráulico"), el sistema siguió confundiendo fluidos, recuperando advertencias sobre Aceite de Motor.
- Barrera de Idioma Persistente: Las consultas en español sobre manuales en inglés (ID 20-27) fallaron sistemáticamente, demostrando que el modelo multilingüe no puede realizar el "puente terminológico" (ej. "Tuerca" $\neq$ "Nut") con la precisión necesaria sin traducción previa.

Fase 3: Implementación de Estrategia Híbrida (Vectores + BM25). La Fase 3 combina la búsqueda vectorial con un recuperador léxico basado en BM25 (Robertson, Jones, Zaragoza, & Walker, 2009). Mientras la búsqueda vectorial compara el significado general de la consulta con el de cada fragmento, BM25 puntúa los fragmentos según cuántas de las palabras de la consulta contienen y qué tan poco frecuentes son en el corpus. Por eso ambos métodos se complementan. El vectorial tolera diferencias de redacción, y el léxico favorece la coincidencia exacta de códigos, modelos y nombres de componentes, que son frecuentes en los manuales técnicos.

Ambos recuperadores trabajan sobre los mismos fragmentos generados en la ingesta. Cada uno produce su propia lista ordenada de candidatos y las dos listas se combinan mediante Reciprocal Rank Fusion (Cormack, Clarke, & Büttcher, 2009). Este método asigna a cada fragmento una puntuación según la posición que ocupa en cada lista, de modo que quienes aparecen bien posicionados en ambas quedan primeros. Al basarse en posiciones y no en los puntajes originales, evita comparar valores de naturaleza distinta, como el puntaje de BM25 y la distancia vectorial. Ambos recuperadores contribuyen con el mismo peso.

Sobre la lista combinada se aplica, finalmente, el filtro por modelo y categoría del manual, y se entregan los cinco primeros fragmentos que lo cumplen. La implementación del componente léxico es deliberadamente simple: usa la configuración por defecto de la librería, sin normalización de mayúsculas, acentos ni formas de las palabras. Esto prioriza la coincidencia literal, pero limita su desempeño cuando el texto extraído presenta palabras unidas o artefactos de codificación. Por ello, los resultados reflejan el aporte de esta configuración y no el máximo que podría lograr un componente léxico ajustado.

Justificación de la Complementariedad La Tabla 7 ilustra cómo ambos métodos cubren las deficiencias del otro.

Tabla 7. Comparativa de desempeño por tipo de algoritmo.

Tipo de Consulta	Desempeño Vectorial (Semántico)	Desempeño BM25 (Léxico)	Resultado Híbrido
Intención Procedimiento <i>(Ej: "¿Cómo bajar la carga?")</i>	Alto. Entiende sinónimos ("bajar" ≈ "descender").	Bajo. Falla sin coincidencia exacta.	Éxito (Vectores).
Datos Exactos / Códigos <i>(Ej: "Aceite 15W-40")</i>	Bajo. Tiende a "alucinar" números similares.	Muy Alto. Encuentra la literalidad.	Éxito (BM25).
Terminología Específica <i>(Ej: "Riel", "Oruga")</i>	Variable. Depende del entrenamiento.	Alto. Si el manual usa el término exacto.	Robusto.

Resultados Finales de la Validación La combinación de estrategias generó el salto de rendimiento más significativo del estudio (Tabla 8).

Tabla 8. Matriz de resultado método híbrido

Categoría de Prueba	Total	Hits Reales	Falsos Positivos	Miss	HitRate
Seguridad Crítica	9	4	3 (ID 3,7,8)	2	44.4%
Operación (Español)	10	7	3 (ID 11, 12, 13)	0	70.0%
Mantenimiento (Inglés)	8	1	1 (ID 20)	6	12.5%
Troubleshooting	3	1	0	2	33.3%
GLOBAL	30	13	7	10	43.3%

Análisis de Casos de Éxito (Corrección de Ambigüedad)

- Corrección del Caso ID 1 (Extintor): Al utilizar BM25 en conjunto con vectores, el sistema logró anclar la búsqueda al término específico "cabina" y "extintor", filtrando el ruido de "incendio" y recuperando el fragmento correcto (Rank 1).
- Éxito en Procedimientos Complejos (Caso ID 5): La consulta sobre "Barra de bloqueo" logró un emparejamiento perfecto gracias a la unicidad del término técnico, eliminando la ambigüedad general de "dirección".

Resultados Finales: La combinación de estrategias generó el salto de rendimiento más significativo, alcanzando un HitRate Global del 43.3%.

- Impacto en Seguridad: La recuperación de elementos críticos (ej. Extintores) subió al 44.4%, validando que la búsqueda léxica (BM25) con enriquecimiento de consultas, es fundamental para anclar la búsqueda a términos físicos inequívocos.

Conclusión de la Validación La arquitectura híbrida corrige significativamente los problemas de ambigüedad y recuperación errónea en consultas en español. Sin embargo, el bajo desempeño en documentación en inglés (12.5%) identifica la necesidad imperativa de un módulo de traducción o un modelo Cross-Encoder para futuras iteraciones, los casos críticos discutidos, se encuentran en el Anexo A: Casos críticos.

Caracterización Estadística e Intervalos de Confianza: Con el propósito de caracterizar la incertidumbre asociada a la muestra de $N = 30$ consultas y evitar interpretaciones sustentadas únicamente en valores puntuales, se incorporan Intervalos de Confianza (IC al 95 %) para la proporción de éxito ($\hat{p}$) de cada estrategia. Estos intervalos acompañan al HR@5 y permiten evaluar, además, si el desempeño observado es compatible con el umbral de aceptación definido en la sección 3.2.

Dada la naturaleza dicotómica de la variable de respuesta Y_i en muestras acotadas ($N < 100$), se aplica la estimación por el *Método de Wilson Score*:

$$IC_{95\%} = \frac{\hat{p} + \frac{z^2}{2N} \pm z \sqrt{\frac{\hat{p}(1-\hat{p})}{N} + \frac{z^2}{4N}}}{1 + \frac{z^2}{N}}$$

Donde $z = 1.96$ para un nivel de significancia $\alpha = 0.05$.

Tabla 9. Caracterización estadística de las estrategias evaluadas ($N = 30$)

Estrategia de Recuperación (X)	Aciertos (N_h)	Falsos Positivos (FP)	Estimación Puntual ($\hat{p}$)	IC 95% (Wilson)	Latencia Promedio (L)
X_1 : Vectorial Pura (Línea Base)	8	8	26.7%	[14.2%,44.4%]	18ms
X_2 : Vectorial Enriquecida	11	4	36.67%	[21.9%,54.5%]	21ms
X_3 : Ensamble Híbrido (Vectores + BM25)	13	7	43.33%	[27.4%,60.8%]	23ms

Nota. X_3 combina varios cambios respecto a X_1 (consulta enriquecida, recuperación híbrida, adición del modelo y la categoría al texto de la consulta y filtrado sobre un conjunto ampliado de candidatos). Su descripción detallada se encuentra al inicio de la Fase 3.

Análisis de Variabilidad: El cálculo de los intervalos de confianza visibiliza la dispersión intrínseca de la muestra: cada intervalo abarca cerca de 30 puntos porcentuales, por lo que el desempeño real de cada estrategia podría diferir de forma importante del valor observado. Aunque la estrategia final (X_3) incrementa el HR@5 en 16.7 puntos porcentuales respecto a la línea base (X_1), los intervalos de ambas se superponen parcialmente ([14.2 %, 44.4 %] y [27.4 %, 60.8 %]). Esta superposición se interpreta solo como indicio de incertidumbre; la comparación formal entre estrategias se realiza

mediante la prueba de McNemar para datos pareados. La variabilidad observada obliga a calificar los resultados con cautela e indica la necesidad de evaluar corpus extendidos en trabajos futuros.

Contraste con el Umbral de Aceptación: En la sección 3.2 se estableció un umbral de aceptación de $HR@5 \geq 0.7$ como requisito para considerar funcional de manera autónoma al motor de búsqueda. Para la estrategia final (X_3), el límite superior del IC al 95 % (60.8 %) es inferior a dicho umbral. Se aplicó además una prueba binomial exacta unilateral con $H_0: p \geq 0.7$ frente a $H_1: p < 0.7$:

$$P(X \leq 13 \mid n = 30, p = 0.7) = \sum_{k=0}^{13} C(30, k)(0.7)^k(0.3)^{30-k} \approx 0.002$$

Se rechaza H_0 al nivel de 0.05. En consecuencia, con este corpus y este conjunto de consultas, el módulo de recuperación no alcanza el criterio de operación autónoma, y esta conclusión se mantiene aun considerando la incertidumbre muestral.

Prueba de Hipótesis Pareada: Prueba de McNemar: Para determinar si el conjunto de modificaciones incorporadas en la estrategia final (X_3) genera un cambio estadísticamente significativo en la clasificación de aciertos frente a la línea base vectorial (X_1), se formula la Prueba de McNemar para muestras dependientes sobre las mismas 30 consultas. Dado que X_3 combina varios factores, el contraste evalúa su efecto conjunto y no el de un componente aislado.

Formulación de Hipótesis

- Hipótesis Nula (H_0): No existe diferencia en la tasa de aciertos entre la recuperación vectorial con consulta original (X_1) y la estrategia final (X_3) ($p_{01} = p_{10}$).
- Hipótesis Alternativa (H_1): La estrategia final (consulta enriquecida, recuperación híbrida, términos de refuerzo y filtrado ampliado) modifica la tasa de aciertos respecto a la línea base ($p_{01} \neq p_{10}$).

Tabla 10. Matriz de Contingencia Pareada (2x2)

	X_3 Híbrido: Acierto (Hit)	X_3 Híbrido: Fallo (Miss)
X_1 Vectorial: Acierto (Hit)	$a = 7$ (Aciertos concordantes)	$b = 1$ (Acierta X_1 , falla X_3)
X_1 Vectorial: Fallo (Miss)	$c = 6$ (Falla X_1 , corrige X_3)	$d = 16$ (Fallos concordantes)

Cálculos Estadísticos

Estadístico de McNemar con Corrección de Yates:

$$\chi^2 = \frac{(|b - c| - 1)^2}{b + c} = \frac{(|1 - 6| - 1)^2}{1 + 6} = \frac{16}{7} \approx 2.2857 \rightarrow p = 0.1306$$

Prueba Binomial Exacta (recomendada para $b + c = 7 < 20$):

$$p_{\text{exacto}} = 2 \times \sum_{k=0}^1 \binom{7}{k} (0.5)^7 = 2 \times \frac{1+7}{128} = \frac{16}{128} = 0.1250$$

Interpretación Metodológica: Desde una perspectiva estrictamente estadística, $p = 0.1250 > 0.05$ impide rechazar H_0 al nivel de confianza del 95 % para este tamaño muestral acotado ($N = 30$). De las 7 consultas en que ambas estrategias difieren, 6 favorecen a la estrategia final y 1 a la línea base, lo que se traduce en 5 aciertos adicionales. Este resultado indica evidencia insuficiente para afirmar una mejora, pero no equivale a demostrar la ausencia de efecto. Asimismo, dado que X_3 combina varios cambios, la mejora observada no puede atribuirse a ninguno de ellos en particular, ni al enriquecimiento de las consultas ni al componente léxico BM25.

Conclusión Estadística: El análisis respalda dos conclusiones: (i) no hay evidencia suficiente para afirmar que la estrategia final supere a la línea base en este conjunto de consultas; y (ii) el desempeño de la estrategia final se ubica por debajo del umbral de 70 %, incluso considerando la incertidumbre de la muestra.

Análisis de Amenazas a la Validez

Validez Interna

- Sesgo de Formulación de Consultas: Derivado de la elaboración y expansión manual de las 30 preguntas de prueba.
- Composición de la estrategia final: X_3 difiere de la línea base en varios factores simultáneos, por lo que el diseño no permite aislar el aporte de cada uno. En particular, no se evaluó BM25 como línea base independiente y su configuración fue la predeterminada de la librería (sin normalización de mayúsculas, acentos ni formas de las palabras).
- Brecha Lingüística: Se identificó una degradación severa en la tasa de recuperación al evaluar consultas en español sobre manuales redactados en inglés, alcanzando un HR@5 de apenas 12.5 %. Este resultado es consistente con limitaciones en la proyección del espacio vectorial del modelo multilingüe al procesar vocabulario técnico especializado, pero no permite aislar el efecto del idioma de otros factores: el tipo de manual, la calidad de la extracción de texto y el filtrado posterior a la recuperación, que deja la lista vacía cuando ningún candidato pertenece al manual esperado (seis de las ocho consultas de esta categoría terminaron sin resultados).

Validez Externa

Acotamiento del Corpus: La evaluación se acotó a un corpus de 5 manuales de un único fabricante industrial (Liebherr, totalizando 4.352 fragmentos vectorizados). Las consultas constituyen una muestra de conveniencia y no aleatoria, por lo que los intervalos de confianza son válidos únicamente para este conjunto. Además, todas las consultas se evaluaron indicando al sistema el modelo y la categoría del manual esperado; el HR@5 corresponde, por tanto, a un escenario en que el usuario conoce el manual consultado. La generalización de estos hallazgos a otros formatos o dominios documentales debe realizarse con resguardo.

Validez de Constructo

Representatividad del Indicador HR@5: La métrica evalúa únicamente si el fragmento objetivo figura entre los 5 primeros pasajes recuperados. No obstante, esto no garantiza que el operario pueda resolver la contingencia operativa, pues un fragmento temáticamente cercano pero no pertinente puede inducir a error. Por ello, los aciertos preliminares del sistema (lista de resultados no vacía) se reclasificaron manualmente, y los falsos positivos se reportan de forma separada. La utilidad de una capa de síntesis o razonamiento generativo contextual (RAG) sobre estos fragmentos no fue evaluada en este trabajo.

Contribución potencial al proceso de consulta de documentación técnica: La solución propuesta busca facilitar la localización de información relevante en manuales técnicos mediante búsqueda semántica, reduciendo la dependencia de búsquedas basadas exclusivamente en coincidencias léxicas. La latencia observada evidencia que el proceso de recuperación puede ejecutarse en tiempos reducidos bajo las condiciones evaluadas. La latencia corresponde únicamente a la etapa de recuperación sobre un corpus de 4.352 fragmentos y no es comparable con la de sistemas que incluyen generación de respuestas, sin embargo, el estudio no mide directamente una reducción del tiempo de diagnóstico, del tiempo medio de reparación ni de los costos operacionales. Estos efectos constituyen beneficios potenciales que requieren una evaluación posterior en un escenario de uso industrial.

Posicionamiento frente a trabajos relacionados

La solución propuesta se ubica en la etapa de recuperación de información, no en la de generación de respuestas. Por ello, la comparación con trabajos relacionados se plantea a nivel de decisiones de diseño y no de desempeño, ya que los trabajos revisados en la sección 2.4 se evaluaron en corpus, métricas y condiciones distintas a las de este estudio, y no se dispone de una medición propia bajo condiciones equivalentes.

Frente a los esquemas de recuperación densa que suelen operar como primera etapa de una arquitectura RAG, la propuesta se diferencia en dos aspectos. Primero, combina búsqueda vectorial con un índice léxico (BM25), orientado a capturar coincidencias literales de códigos y terminología técnica. Segundo, se ejecuta íntegramente en entorno local, con un modelo de embeddings de código abierto y un índice FAISS, sin depender de servicios externos en la etapa de recuperación. Esto es coherente con las directrices definidas en la sección 2.3 (ejecución local y baja complejidad de configuración), aunque no se midió su efecto frente a alternativas desplegadas en la nube.

Frente a enfoques jerárquicos como HiQA (Chen et al., 2024) y a los basados en grafos de conocimiento (Hogan et al., 2021), la propuesta adopta una arquitectura deliberadamente más simple. Como se discutió en la sección 2.4, esos enfoques ofrecen ventajas teóricas para preservar relaciones estructurales y responder consultas de múltiples saltos, pero exigen procesos de extracción de entidades y relaciones más costosos. La decisión de este trabajo prioriza la facilidad de implementación y la actualización incremental del índice, a cambio de no capturar dependencias jerárquicas entre componentes. Esta limitación se refleja en los resultados, ya que los fallos observados incluyen consultas sobre procedimientos y relaciones entre elementos que un recuperador de fragmentos independientes no representa.

Una diferencia relevante es que la precisión de este módulo es limitada: el sistema híbrido alcanzó un HitRate@5 de 43,3 % (IC 95 %: 27,4 %–60,8 %), por debajo del umbral de 70 % definido como requisito de operación autónoma. Por esta razón, las ventajas de la propuesta en infraestructura y latencia (inferior a 25 ms en la recuperación sobre 4.352 fragmentos) no deben interpretarse como superioridad sobre otros enfoques. Dicha latencia corresponde únicamente a la etapa de recuperación sobre un corpus acotado, y no es comparable con la de sistemas que incluyen generación con modelos de lenguaje. Si bien las arquitecturas más complejas podrían lograr mayor precisión, esta hipótesis no fue evaluada aquí, y comparar su desempeño bajo las mismas consultas queda como trabajo futuro, junto con la incorporación de una línea base léxica (BM25 y búsqueda por palabras clave).

Criterio	Solución propuesta	Recuperación densa como etapa de RAG	Enfoques jerárquicos o basados en grafos
Rol en el sistema	Módulo de recuperación	Primera etapa antes de la generación	Recuperación estructurada, a veces con generación
Mecanismo de búsqueda	Vectorial (FAISS) + léxico (BM25)	Vectorial; puede incorporar variantes híbridas	Jerarquías o recorrido de grafos
Despliegue del módulo	Local, desacoplado	Depende de la implementación	Depende de la implementación
Costo de preparación del corpus	Moderado (extracción, chunking, embeddings)	Similar	Mayor (extracción de entidades y relaciones)
Actualización con nuevos documentos	Incremental sobre el índice	Re-indexación vectorial	Más compleja según la literatura revisada
Representación de relaciones estructurales	No incluida	No incluida	Sí, de forma explícita
Latencia	< 25 ms (solo recuperación, 4.352 fragmentos)	No comparable	No comparable
Calidad de recuperación	HR@5 = 43,3 % en 30 consultas	No evaluada en este estudio	No evaluada en este estudio

Integración con Otros Componentes

Más allá del núcleo de procesamiento vectorial, se diseñó una arquitectura de software orientada a servicios que permite la integración fluida con aplicaciones externas y garantiza la consistencia de los datos en entornos dinámicos. Esta estructura asegura que el módulo de búsqueda no sea un componente aislado, sino un servicio consumible por el ecosistema digital de la organización.

Extensibilidad del Módulo de Búsqueda. El sistema se organiza en una arquitectura de tres niveles que desacopla la lógica de recuperación de la interfaz de usuario, facilitando la escalabilidad y el mantenimiento independiente de los componentes.

- Capa de Presentación (Clientes): Corresponde a las aplicaciones externas (Chatbots, Web Apps, Dispositivos Móviles) que requieren acceso a la información técnica. Estas interactúan con el sistema exclusivamente a través de protocolos estándar (HTTP/JSON).
- Capa de Aplicación (API Gateway): Implementada sobre el framework Django (archivo `views.py`), actúa como puerta de entrada única. Expone endpoints REST (`/ask`, `/search`) que validan las peticiones, gestionan la autenticación y orquestan las llamadas a los servicios internos.
- Capa de Dominio (Core Services):
 - Módulo RAG (`llm_service.py`): Gestiona la interacción con el Modelo de Lenguaje, la construcción del prompt y el refinamiento de la respuesta.
 - Servicio Vectorial (`vector_service.py`): Administra las operaciones de bajo nivel sobre el índice FAISS, encapsulando la complejidad de la vectorización y la búsqueda por similitud.

Interfaces de Integración (Patrón Fachada). Para simplificar el consumo de los recursos, el sistema expone interfaces diferenciadas según el perfil del consumidor:

- Perfil Documental: Permite el acceso directo a los fragmentos (chunks) recuperados sin procesamiento generativo, útil para sistemas de auditoría o visualizadores de manuales que requieren la fuente cruda.
- Perfil Asistente: Habilita el flujo completo RAG, retornando respuestas sintetizadas en lenguaje natural acompañadas de citas y advertencias de seguridad.
- Perfil Analítico: Provee acceso a la clase `VectorDBMonitor` para la extracción de telemetría, permitiendo a los administradores visualizar métricas de cobertura y rendimiento del índice en tiempo real.

Estrategia de Sincronización Incremental. Para asegurar que la base de datos vectorial permanezca vigente y consistente sin requerir costosas reconstrucciones completas, se implementó un algoritmo de Sincronización Diferencial en el módulo `vector_service.py`. Este proceso se basa en la detección de deltas (cambios) entre el estado actual de los manuales en la base de datos y el registro histórico del índice (`processed_pdfs.json`).

Detección y Purga de Eliminaciones (Consistencia) Esta fase garantiza que la información obsoleta sea removida del índice FAISS, manteniendo la consistencia de los datos en tiempo de consulta.

El mecanismo utiliza la teoría de conjuntos para identificar los IDs que deben ser eliminados, comparando el registro guardado con el snapshot actual (Figura 10)

```
current_snapshot = build_processed_manuals_snapshot()  
removed_manual_ids = sorted(set(processed_manuals.keys()) -  
set(current_snapshot.keys()))
```

Figura 10. Lógica de detección de IDs eliminados.

Si se detectan IDs eliminados, se remueven del índice en memoria, preservando los recursos computacionales (Figura 11).

```
if ids_to_delete:  
    if hasattr(vectorstore, "delete"):  
        try:  
            vectorstore.delete(ids_to_delete)  
            vectorstore_dirty = True  
            logger.info(f"Eliminados {len(ids_to_delete)} vectores de  
manuales borrados")  
        except Exception as exc:  
            logger.exception(f"No se pudieron eliminar vectores  
obsoletos: {exc}")  
            vectorstore = None
```

Figura 11. Limpieza de vectores de manuales eliminados.

El bloque try-except aquí es clave para gestionar fallos críticos (si FAISS no soporta la eliminación directa) y forzar la reconstrucción del índice (vectorstore = None) para prevenir inconsistencias.

Detección Condicional de Procesamiento Esta fase determina la eficiencia del sistema, asegurando que el costoso proceso de chunking y embedding solo se ejecute cuando es estrictamente necesario.

La función `get_manuals_to_process` (Figura 12) identifica los documentos que son Nuevos (su ID no está en el histórico) o Modificados (el identificador de archivo en Django no coincide con el registro anterior).

```
Def get_manuals_to_process(existing_pdfs: dict[str, str]) ->
list[Manual]:
    manuals_to_process = []
    for manual in _manual_queryset():
        storage_name = manual.file.name if manual.file else None
        if not storage_name:
            continue
        manual_id = str(manual.id)
        # Condición: Si el ID es nuevo O el nombre del archivo (hash
        implícito) cambió
        if manual_id not in existing_pdfs or existing_pdfs[manual_id] !=
        storage_name:
            if resolve_file_path(manual):
                manuals_to_process.append(manual)
    return manuals_to_process
```

Figura 12. Función de procesamiento de manuales.

La lógica principal solo procede si el índice debe crearse desde cero o si hay manuales pendientes de procesar (Figura 13).

```
if not vectorstore or manuals_to_process:
    if not vectorstore:
        logger.info("📁 Creando un nuevo índice desde cero...")
        # ... lógica de Cold Start ...
    else:
        logger.info(f"📁 Agregando {len(manuals_to_process)} nuevos
        documentos...")
        # ... lógica de Adición Incremental ...
```

Figura 13. Condicional para optimización de recursos.

Inserción Incremental y Persistencia si se detectan nuevos o modificados, se procesan, se añaden al índice existente, y se persiste el estado para cerrar el ciclo de sincronización.

Adición Incremental: Los nuevos documentos son añadidos al índice cargado en memoria (Figura 14).

```
# Adición de nuevos documentos al índice existente
if new_docs:
    vectorstore.add_documents(new_docs)
    vectorstore_dirty = True
    logger.info(f"{len(new_docs)} nuevos documentos agregados al índice")
```

Figura 14. Logica de adición de documentos nuevos al indice.

Persistencia de Estado: El índice FAISS se guarda en disco y el registro de manuales procesados se actualiza, asegurando que el sistema "recuerde" el estado del archivo para el siguiente ciclo (Figura 15).

```
if vectorstore and vectorstore_dirty:
    vectorstore.save_local(FAISS_INDEX_PATH)
    logger.info("Índice FAISS guardado")

if processed_registry_dirty:
    # Actualiza el archivo processed_pdfs.json
    latest_snapshot = build_processed_manuals_snapshot()
    with open(PROCESSED_PDFS_FILE, 'w', encoding='utf-8') as f:
        json.dump(latest_snapshot, f)
```

Figura 15. Guardado de Índice FAISS y registro de estado.

Esta secuencia de pasos garantiza que el índice FAISS esté siempre vigente y que el sistema sea eficiente en entornos productivos con alta volatilidad documental.

4 Conclusiones

El presente trabajo de tesina cumplió el objetivo de diseñar e implementar un módulo de recuperación semántica para manuales técnicos y lo caracterizó empíricamente. Los objetivos de extracción, vectorización, diseño de la base de datos y documentación se cumplieron. El objetivo de evaluación se cumplió de forma parcial: el módulo no alcanzó el umbral de precisión definido y no se contrastó con una línea base léxica independiente.

En el plano de la ingeniería de datos, el proyecto implementó un pipeline capaz de extraer contenido de manuales en formato PDF mediante una estrategia híbrida (texto digital y OCR), vectorizarlo con representaciones densas y organizarlo en una arquitectura modular. En las condiciones evaluadas (4.352 fragmentos y ejecución local), la recuperación presentó una latencia promedio de entre 18 y 23 ms según la estrategia, inferior al SLA de 25 ms definido para este prototipo. Esta latencia corresponde únicamente a la etapa de recuperación, y la escalabilidad a volúmenes mayores no fue evaluada. El módulo incorpora además un mecanismo de sincronización incremental del índice, diseñado para facilitar su mantenimiento, cuya operación no formó parte de la validación.

En cuanto al objetivo de evaluación, la validación en tres fases permitió caracterizar el desempeño del módulo sobre 30 consultas. La estrategia final, que combina consultas enriquecidas, recuperación híbrida (vectorial y BM25) y términos de refuerzo, obtuvo el mayor HitRate@5 (43,3 %, frente a 26,7 % de la línea base). Sin embargo, la prueba de McNemar no detectó una diferencia estadísticamente significativa respecto a la línea base ($p = 0,125$). Por tanto, no hay evidencia suficiente para afirmar una mejora, y el diseño no permite atribuirle a ningún componente en particular.

El resultado central es que el nivel de acierto no alcanza el umbral de 0,7 definido como requisito de operación autónoma. El HitRate@5 de la estrategia final fue de 43,3 % (IC 95 %: 27,4 %-60,8 %), con un límite superior inferior al umbral (prueba binomial exacta, $p \approx 0,002$). Este resultado es válido para este corpus y este conjunto de consultas, evaluadas indicando al sistema el manual esperado. La validación tiene limitaciones: muestra de conveniencia, un único revisor y ausencia de una línea base léxica independiente. Estos resultados motivan evaluar capas adicionales sobre el módulo, pero no permiten asumir su efecto.

Por lo tanto, el módulo de recuperación resultante es técnicamente viable y eficiente en el corpus evaluado, pero no satisface por sí solo el criterio de precisión fijado para una operación autónoma. Su aporte dentro del proyecto mayor es entregar una base de recuperación con desempeño caracterizado (HitRate@5 ≈ 43 %), que los módulos posteriores deberán considerar como punto de partida. Se establecen las siguientes directrices como trabajo futuro:

Evolución Cognitiva (Integración RAG): Los resultados obtenidos justifican evaluar la integración de una arquitectura de Generación Aumentada por Recuperación (RAG) como una posible extensión del sistema. Una capa generativa depende de la calidad del recuperador: en cerca del 57 % de las consultas el fragmento relevante no figuró entre los cinco primeros, y en parte de ellas se recuperaron fragmentos plausibles pero incorrectos. Por ello, su efecto no puede asumirse y su diseño debería incluir mecanismos de abstención y advertencia. Su efectividad deberá validarse mediante experimentos específicos.

Resolución de la Barrera Idiomática: Las consultas en español sobre manuales redactados en inglés alcanzaron un HitRate@5 de 12,5 % ($n = 8$) y la mayoría terminó sin resultados. Esto sugiere evaluar una traducción previa de consultas o documentos, un modelo de recuperación multilingüe adaptado al dominio, o un reordenamiento de candidatos con un modelo cross-encoder multilingüe. El efecto del idioma debería separarse del de la calidad de extracción del texto y del filtrado por manual, que pueden contribuir a este resultado.

Escalabilidad Algorítmica Masiva: El índice de búsqueda exhaustiva actual (IndexFlatL2) es adecuado para el corpus evaluado. Si el volumen de vectores escalara a cientos de miles, podría ser necesario migrar a índices aproximados como IndexIVF (Inverted File Index), evaluando el compromiso entre velocidad y pérdida de precisión. Esta transición no fue probada en este trabajo.

Razonamiento Estructural (Grafos de Conocimiento): Algunos fallos observados en consultas sobre procedimientos secuenciales son consistentes con la dificultad de los fragmentos independientes para representar dependencias entre pasos o componentes. Se proyecta explorar arquitecturas que combinen los vectores con Grafos de Conocimiento (Knowledge Graphs), lo que habilitaría un razonamiento “multisalto” capaz de preservar dependencias jerárquicas, considerando el costo de extracción de entidades y relaciones discutido en la sección 2.4.

Ampliación de la Validación y Línea Base Léxica: Se propone repetir la evaluación con un conjunto mayor de consultas, anotación previa de la relevancia por más de un evaluador y un subconjunto reservado, e incorporar como líneas base la búsqueda léxica BM25 por sí sola y la búsqueda por palabras clave. Esto permitiría contrastar estrategias con mayor potencia estadística y atribuir el aporte de cada componente.

Mejoras del Módulo de Recuperación: Se propone aplicar el filtrado por manual antes de la recuperación y no sobre los resultados, normalizar el texto del índice léxico (mayúsculas, acentos y formas de las palabras), revisar la calidad de la extracción en los manuales con texto deficiente y evaluar la cobertura con un número mayor de candidatos ($k > 5$) para determinar si el fragmento relevante se encontraba en posiciones más profundas.

5 Agradecimientos.

Quisiera agradecer, en primer lugar, a mi familia, que me ha apoyado desde la distancia durante este largo proceso. También a mis cercanos y amigos, quienes han estado presentes tanto en los momentos buenos como en los momentos más difíciles, especialmente a mi amigo Boris, que me acompañó de cerca durante gran parte de este camino.

Por último, quisiera agradecer a quienes ya no están o partieron para siempre. Gracias a ellos soy quién soy hoy; dondequiera que estén, les estaré siempre agradecido.

6 Referencias

- Chen, X., Gao, P., Song, J., & Tan, X. (2024). *HiQA: A Hierarchical Contextual Augmentation RAG for Multi-Documents QA*. Cornell University.
- Cormack, G., Clarke, C., & Büttcher, S. (2009). Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)* (págs. 758-759). ACM.
- Färber, M., Bartscherer, F., Menne, C., & Rettinger, A. (2018). Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web*.
- Gomez-Cabello, C. A., Prabha, S., Haider, S. A., Genovese, A., Collaco, B. G., Wood, N. G., . . . Forte, A. J. (2025). Comparative Evaluation of Advanced Chunking for Retrieval-Augmented Generation in Large Language Models for Clinical Decision Support. *Bioengineering* 2025.
- Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., . . . Neumaier, S. (2021). Knowledge graphs. *ACM Computing Surveys*.
- Hugging Face. (2016). *Hugging Face documentation*. Obtenido de <https://huggingface.co/docs>
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., . . . Yih, W.-T. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- LangChain. (2022). Obtenido de Text splitters: https://python.langchain.com/docs/concepts/text_splitters
- Meta AI. (2017). Obtenido de FAISS documentation: <https://faiss.ai>
- Milvus. (2017). Obtenido de Milvus documentation: <https://milvus.io/docs>
- Mitra, B., & Craswell, N. (2018). An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*.
- OpenAI. (2015). Obtenido de Embeddings API: <https://platform.openai.com/docs/guides/embeddings>
- Pinecone Systems, Inc. (2019). Obtenido de Vector database: <https://www.pinecone.io>
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Robertson, S. E., Jones, K. S., Zaragoza, H., & Walker, S. J. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*.
- Weaviate. (2019). Obtenido de Architecture overview: <https://weaviate.io/developers/weaviate/architecture>
- Xie, Q., Huang, J., Du, P., Zhang, M., & Huang, X. (2022). Hybrid retrieval approach for technical documentation using knowledge graphs and dense vectors. *Information Processing & Management*.

7 Anexos

7.1 Anexo A: Casos críticos

Tabla 11. Casos Críticos Fase 1 (Consultas Crudas)

ID	Consulta	Relevancia	Observación
1	¿Qué tipo de extintor se debe utilizar en caso de incendio eléctrico y dónde está ubicado?	No relevante	Los fragmentos hablan de sensores de alarma y trabajos de mantenimiento, o la posibilidad de colocar un extintor en la cabina. Sin embargo, ninguno menciona el tipo de agente extintor (necesario para fuegos eléctricos).
12	¿Qué significa el testigo luminoso de temperatura de aceite hidráulico?	No relevante	El único fragmento recuperado se refiere a la indicación de temperatura muy alta en la transmisión del sistema de traslación, que es aceite de engranajes, no el aceite hidráulico de trabajo (tanque principal). No se identifica el testigo luminoso del aceite hidráulico.
17	¿Cuál es el patrón de control ISO para los joysticks izquierdo y derecho?	No relevante	Los fragmentos hablan de conmutación de movimientos hidráulicos adicionales con el interruptor basculante S55 en la palanca de mando en cruz y describen algunas funciones de botones (giro de la almeja). No se presenta un esquema claro o la descripción completa del patrón ISO/BHL de los movimientos principales (Pluma/Balancín/Cuchara/Giro) en los joysticks izquierdo y derecho.

Tabla 12. Casos Críticos Fase 2 (Consultas Enriquecidas)

ID	Consulta	Relevancia	Observación
1	Especificación y ubicación del extintor de incendios portátil para fuegos eléctricos en cabina. Tipo de agente extintor y montaje en excavadora Liebherr.	No relevante	Los fragmentos hablan de: trabajar con la puerta abierta, distancias de seguridad para líneas eléctricas, sensores de temperatura de alarmas de incendio, y llenado de combustible. Ninguno menciona la ubicación, el tipo (agente) o la especificación (clase C/eléctricos) del extintor portátil.
12	Significado y acción requerida para el símbolo de advertencia o testigo luminoso de sobrecalentamiento del aceite hidráulico en el panel de instrumentos. Icono termómetro.	No relevante	El único fragmento recuperado describe la simbología de combustible, presión de aceite de motor y la temperatura del líquido refrigerante. No hay mención específica al sobrecalentamiento del aceite hidráulico ni al icono asociado.

Tabla 13. Casos Críticos Fase 3 (Híbrido semántico+léxico)

ID	Consulta	Relevancia	Observación
1	Especificación y ubicación del extintor de incendios portátil para fuegos eléctricos en cabina. Tipo de agente extintor y montaje en excavadora Liebherr.	Relevante	Los fragmentos indican que existe la posibilidad de colocar un extintor en la cabina por medio de 4 agujeros roscados situados en el larguero izquierdo de la parte trasera de la misma.
5	Instalación de la barra de bloqueo de la articulación de dirección para aseguramiento durante transporte, izaje o mantenimiento. Seguro de articulación.	Relevante	Se hace referencia al "Bloqueo de la articulación" en el índice, y se advierte del peligro de accidente si se intenta dirigir con el bloqueo puesto.

7.2 Anexo B: Resultado detallados

Tabla 14. Resultados de validación

ID	Categoría	Vectorial Base	V-Enriquecida	Híbrida
1	Seguridad	✗ MISS(FP)	✗ MISS(FP)	✓ HIT
2	Seguridad	✓ HIT	✓ HIT	✓ HIT
3	Seguridad	✗ MISS(FP)	✗ MISS(FP)	✗ MISS(FP)
4	Seguridad	✗ NO	✗ NO	✗ NO
5	Seguridad	✗ MISS(FP)	✓ HIT	✓ HIT
6	Seguridad	✗ NO	✗ NO	✗ NO
7	Seguridad	✗ NO	✗ NO	✗ MISS(FP)
8	Seguridad	✗ NO	✓ HIT	✗ MISS(FP)
9	Seguridad	✓ HIT	✓ HIT	✓ HIT
10	Operación	✗ NO	✗ NO	✓ HIT
11	Operación	✗ NO	✗ NO	✗ MISS(FP)
12	Operación	✗ MISS(FP)	✗ MISS(FP)	✗ MISS(FP)
13	Operación	✗ NO	✗ NO	✗ MISS(FP)
14	Operación	✓ HIT	✓ HIT	✓ HIT
15	Operación	✓ HIT	✓ HIT	✓ HIT
16	Operación	✓ HIT	✓ HIT	✓ HIT
17	Operación	✗ MISS(FP)	✓ HIT	✓ HIT
18	Operación	✓ HIT	✓ HIT	✓ HIT
19	Operación	✓ HIT	✓ HIT	✓ HIT
20	Mant (EN)	✗ MISS(FP)	✗ NO	✗ MISS(FP)
21	Mant (EN)	✓ HIT	✗ NO	✗ NO
22	Mant (EN)	✗ NO	✓ HIT	✗ NO
23	Mant (EN)	✗ NO	✗ NO	✗ NO
24	Mant (EN)	✗ NO	✗ NO	✗ NO
25	Mant (EN)	✗ NO	✗ NO	✗ NO
26	Mant (EN)	✗ NO	✗ NO	✗ NO
27	Mant (EN)	✗ MISS(FP)	✗ NO	✓ HIT
28	Troubleshooting	✗ NO	✗ NO	✗ NO
29	Troubleshooting	✗ NO	✗ NO	✗ NO
30	Troubleshooting	✗ MISS(FP)	✗ MISS(FP)	✓ HIT
	RH@5	26.7%	36.7%	43.3%

Nota: "MISS (FP)" son falsos positivos y "NO" son búsquedas que no obtuvieron resultado