UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

Sistema inteligente de búsqueda y coincidencia basado en Vector Stores

LINA MARCELA BUITRAGO ROA

Dirigido por

NICOLÁS COCA LÓPEZ

2025

TÍTULO: SISTEMA INTELIGENTE DE BÚSQUEDA Y COINCIDENCIA BASADO EN VECTOR STORES

AUTOR: LINA MARCELA BUITRAGO ROA

TITULACIÓN: MASTER UNIVERSITARIO EN ANALISIS DE DATOS MASIVOS (BIG DATA)

DIRECTOR/ES DEL PROYECTO: NICOLÁS COCA LÓPEZ

FECHA: Septiembre de 2025

RESUMEN

Esta tesis aborda la brecha que impide a las PYMEs acceder a búsqueda semántica y multimodal de nivel empresarial. Se propone y valida una arquitectura de referencia open-source basada en PostgreSQL + pgvector que integra búsqueda léxica (BM25), vectorial (HNSW), fusión por RRF y RAG-to-SQL orquestado con LangChain. Se emplean embeddings de texto (E5, JE3,GTE) e imagen (CLIP) y se evalúa el sistema sobre una muestra operativa (≈15.000 ítems) del dataset FooDI-ML, utilizando etiquetas estructurales como proxy de relevancia.

El plan de evaluación combina métricas proxy-based (1-NN, Silhouette, ARI/NMI, Label Consistency@K), multimodales (Recall@K, MRR, separación de pares) y de búsqueda híbrida (Hybrid Recall y Filter-Separation). Los resultados muestran que E5 ofrece mejor estructura local y Recall@K con filtros, mientras que GTE/JE3 aportan mayor separación útil para re-ranking. En multimodal, la dirección imagen→texto supera a texto→imagen. La solución opera en CPU en tiempo real (la GPU se limita al backfill offline), manteniendo costes bajos.

Se concluye que la arquitectura propuesta es técnica y económicamente viable para PYMEs y se aporta una “receta” operativa (candidatos híbridos,fusión RRF, re-ranking ligero, multimodal, RAG-to-SQL), junto con líneas futuras.

Palabras clave: Búsqueda híbrida; pgvector; RAG-to-SQL; Recuperación multimodal; HNSW; PYMEs.

ABSTRACT

This thesis tackles the adoption gap that prevents SMEs from using enterprise-grade semantic and multimodal search. It proposes and validates an open-source reference architecture built on PostgreSQL + pgvector, combining lexical search (BM25), vector search (HNSW), rank fusion via RRF, and RAG-to-SQL orchestrated with LangChain. Text (E5, JE3, GTE) and image (CLIP) embeddings are integrated and evaluated on an operational sample (~15k items) from the FooDI-ML dataset, using structured fields as relevance proxies.

The evaluation plan mixes proxy-based metrics (1-NN, Silhouette, ARI/NMI, Label Consistency@K), multimodal metrics (Recall@K, MRR, positive/negative pair separation), and hybrid search metrics (Hybrid Recall, Filter-Separation). Results indicate E5 yields stronger local structure and higher Recall@K under filters, while GTE/JE3 provide separation signals beneficial for re-ranking. For cross-modal retrieval, image→text consistently outperforms text→image. The system meets real-time CPU operation, reserving GPUs for offline embedding backfill, keeping costs low.

Overall, the architecture proves technically and economically feasible for SMEs and delivers a practical recipe (hybrid candidates → RRF fusion → lightweight re-ranking → multimodal → RAG-to-SQL), plus future work.

Keywords: Hybrid search; pgvector; RAG-to-SQL; Multimodal retrieval; HNSW; SMEs.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1. Introducción

1.1 Estado del arte

La recuperación de información ha experimentado una profunda transformación, transitando desde enfoques léxicos, basados en la coincidencia de palabras clave, hacia paradigmas semánticos y multimodales. Esta evolución no solo ha sido impulsada por avances algorítmicos, sino también por una convergencia con nuevas tecnologías de infraestructura, como las bases de datos vectoriales, el almacenamiento distribuido en buckets y los sistemas NoSQL. Esta sinergia ha democratizado el acceso a capacidades avanzadas de búsqueda, abriendo un horizonte de aplicación para las Pequeñas y Medianas Empresas (PYMEs) que va más allá de las soluciones tradicionalmente reservadas a las grandes corporaciones.

La literatura académica documenta esta transición en múltiples frentes. El punto de partida son los métodos léxicos canónicos como BM25 (Robertson & Zaragoza, 2009). La irrupción de los modelos transformer marcó un punto de inflexión, dando lugar a representaciones vectoriales densas con modelos como SBERT (Reimers & Gurevych, 2019) y sus variantes multilingües más recientes como Jina v3 (Jina AI, 2024). Paralelamente, se exploraron arquitecturas más sofisticadas para mejorar la precisión, como los modelos de interacción tardía ColBERT (Khattab & Zaharia, 2020) y las técnicas de re-clasificación (re-ranking) basadas en cross-encoders (Nogueira & Cho, 2019).

El alcance de la búsqueda semántica se expandió al dominio visual con la llegada de modelos multimodales como CLIP (Radford et al., 2021), capaces de crear un espacio de representación unificado para texto e imágenes. La evaluación rigurosa y estandarizada de todos estos enfoques ha sido posible gracias a benchmarks de referencia como BEIR (Thakur et al., 2021).

Desde la perspectiva de la infraestructura, la viabilidad de estas técnicas en entornos productivos depende de dos avances clave. Por un lado, los algoritmos de búsqueda de vecinos próximos aproximados (ANN), como HNSW (Malkov & Yashunin, 2018), que permiten consultas eficientes a escala. Por otro, la integración de estas capacidades en sistemas de datos existentes, ejemplificada por extensiones como pgvector para PostgreSQL (pgvector team, 2024). Esta tendencia, combinada con la flexibilidad de la ingesta de datos desde

almacenamientos distribuidos (ej. S3) y bases de datos no relacionales, ha configurado un ecosistema tecnológico más accesible y escalable.

La aplicación exitosa de estos componentes se observa en casos de uso industriales. Implementaciones a gran escala como la de Walmart (Magnani et al., 2024) demuestran el poder de los sistemas híbridos (léxico + vectorial) que fusionan resultados con métodos como Reciprocal Rank Fusion (RRF) (Cormack et al., 2009). Si bien estas soluciones a medida son costosas, el ecosistema de código abierto actual permite replicar su arquitectura fundamental, lo que representa una oportunidad estratégica para la digitalización de las PYMEs (OECD, 2021).

Esta revisión de la literatura permite identificar un patrón consolidado en los sistemas de búsqueda modernos, que se estructura en las siguientes fases:

- Recuperación inicial de candidatos mediante una combinación de métodos léxicos y densos.
- Fusión de los rankings para maximizar la cobertura y la relevancia inicial.
- Re-clasificación de los resultados más prometedores con modelos de alta precisión.
- Soporte multimodal para integrar señales textuales, visuales y estructuradas.
- Implementación sobre arquitecturas abiertas que combinan búsqueda vectorial y almacenamiento distribuido.

Esta tesis materializa dicho patrón en una solución aplicada, utilizando tecnologías de código abierto (PostgreSQL + pgvector, embeddings públicos) y una arquitectura de datos moderna, con ingesta reproducible desde buckets y formatos columnares (Parquet).

1.2 Contexto y justificación

La capacidad de una organización para explotar sus datos no estructurados se ha convertido en un diferenciador competitivo clave. Mientras las grandes corporaciones tecnológicas han desarrollado ecosistemas de datos avanzados que integran inteligencia artificial para la búsqueda semántica y la personalización, la mayoría de las Pequeñas y Medianas Empresas (PYMEs) aún no han podido realizar esta transición. Esta brecha no es trivial; representa una barrera fundamental para la innovación y la competitividad en el mercado digital actual.

La disparidad se manifiesta claramente en las arquitecturas de datos. Líderes del sector como Amazon (Mohan et al., 2019) y Walmart (Magnani et al., 2024) fundamentan sus operaciones en infraestructuras a gran escala que procesan interacciones complejas. En contraste, un gran número de PYMEs dependen de sistemas transaccionales tradicionales (como MySQL o PostgreSQL) y plataformas CRM, los cuales, si bien son robustos para la gestión operativa, carecen de capacidades nativas para la recuperación de información semántica, visual o conversacional.

Esta limitación tecnológica tiene consecuencias directas y medibles en el negocio. La fricción generada por motores de búsqueda ineficaces deteriora la experiencia del usuario, reduce las tasas de conversión y obstaculiza la fidelización de clientes, un problema documentado por institutos de análisis de la industria (Baymard Institute, 2023). Por tanto, existe una necesidad urgente de desarrollar y validar soluciones que permitan a las PYMEs acceder a tecnologías de búsqueda avanzadas, nivelando el campo de juego competitivo y permitiéndoles capitalizar el valor estratégico de sus propios datos.

1.3 Planteamiento del problema

Si bien las arquitecturas de búsqueda semántica, multimodal y conversacional ya han sido implementadas con éxito a gran escala —por ejemplo, en plataformas empresariales como las de Amazon (Mohan et al., 2019), Walmart (Magnani et al., 2024) y Salesforce (Salesforce, 2024)—, su adopción se basa en infraestructuras propietarias y de alto costo, inaccesibles para la mayoría de las Pequeñas y Medianas Empresas (PYMEs). Esto ha consolidado una brecha tecnológica y competitiva: mientras las grandes corporaciones capitalizan el valor de las búsquedas híbridas y multimodales, las PYMEs continúan dependiendo de motores tradicionales basados en palabras clave, que ofrecen resultados limitados.

El problema central no es, por tanto, la falta de tecnologías, sino la ausencia de una arquitectura de referencia, accesible y validada, que permita a las PYMEs emular estas capacidades avanzadas mediante un ecosistema de código abierto. Los sistemas actuales presentan una fragmentación evidente que impide una solución integral:

- Silos de búsqueda vectorial: Motores semánticos que operan de forma aislada, sin una integración fluida con los datos relacionales donde reside la información transaccional.
- Limitaciones de la búsqueda léxica: Sistemas tradicionales basados en algoritmos como BM25, que son eficientes, pero carecen de comprensión semántica profunda o capacidades multimodales.
- Componentes RAG emergentes: Orquestadores de lenguaje que enriquecen las consultas, pero que aún no resuelven de manera estandarizada la traducción de lenguaje natural a consultas SQL en entornos de datos heterogéneos.

Esta fragmentación tecnológica plantea cuatro desafíos de ingeniería interconectados que deben ser resueltos para ofrecer una solución viable a las PYMEs:

- Integración arquitectónica: La necesidad de unificar datos multimodales (texto, imágenes, datos tabulares) y sus representaciones vectoriales dentro de una única base de datos relacional extendida, como PostgreSQL/pgvector.

- Fusión de señales: El reto de combinar eficazmente los rankings provenientes de métodos léxicos y semánticos, como mediante el algoritmo Reciprocal Rank Fusion (RRF), para maximizar la relevancia.
- Interacción semántica-estructurada: La complejidad de traducir consultas en lenguaje natural, enriquecidas con contexto recuperado semánticamente (RAG), en sentencias SQL correctas y ejecutables.
- Evaluación holística: La falta de un marco que mida simultáneamente la relevancia de la búsqueda, la calidad del SQL generado, la factualidad de las respuestas del RAG y la viabilidad económica de la solución.

En este contexto, la pregunta central de investigación que guía este trabajo se formula de la siguiente manera:

¿Cómo diseñar una arquitectura práctica open-source que integre bases de datos estándar SQL, vectores semánticos y archivos (usando bases vectoriales y almacenamiento tipo S3) para habilitar búsqueda unificada y gestión de objetos enfocado para PYMEs con viabilidad técnica y financiera?

1.4 Objetivos del proyecto

Objetivo general:

Diseñar y validar un motor de recuperación multimodal para datos heterogéneos que combine búsqueda léxica y semántica (texto e imagen) mediante almacenamiento vectorial que integre datos estructurados y representaciones semánticas. La implementación de referencia se realiza en PostgreSQL/pgvector, manteniendo la arquitectura agnóstica a tecnología para facilitar su adopción en otras plataformas equivalentes, con foco en relevancia y eficiencia operativa en pymes.

Objetivos específicos:

1. Diseñar la arquitectura de datos multimodal sobre PostgreSQL/pgvector (esquema, migraciones e índices), manteniendo abstracciones agnósticas para portabilidad a otras plataformas equivalentes.
2. Implementar la ingesta reproducible (S3 → filtrado → Parquet → Postgres) con validaciones por defecto y CLI, asegurando calidad y trazabilidad de datos.
3. Integrar y evaluar embeddings de texto e imagen (JE-3, E5, GTE, CLIP 1024D/512D); realizar backfill eficiente y crear índices HNSW adecuados por espacio vectorial.
4. Construir el motor de búsqueda híbrida (BM25 + vectores) con fusión de rankings (p. ej., RRF) y recuperación multimodal texto-imagen.
5. Implementar un RAG mínimo de demostración e integrar LangChain para generar y ejecutar SQL sobre Postgres, mostrando de forma amigable cómo el contexto recuperado sustenta las respuestas y consultas.
6. Definir y ejecutar un plan de evaluación centrado en:
    - Calidad de SQL: Exact Match, Execution Success y correctitud semántica contra gold sets.
    - Métricas: Score@K, PWR, POC, alignment, hybrid gain
    - Factualidad/Faithfulness del RAG: grado en que las respuestas están sustentadas por el contexto recuperado.
    - Costo operativo: estimación por consulta/sesión (tokens LLM + DB).

Capítulo 2. Marco Teórico

La Recuperación de Información (RI) ha transitado de métodos léxicos, basados en coincidencias literales de términos, hacia paradigmas semánticos y multimodales que buscan capturar el significado y la intención del usuario. Este cambio se sustenta tanto en avances algorítmicos como en innovaciones en infraestructura y aprendizaje profundo (Robertson & Zaragoza, 2009; Vaswani et al., 2017). En este capítulo se presentan los fundamentos teóricos de la búsqueda híbrida y de la Generación Aumentada por Recuperación (RAG), tecnologías que articulan el sistema propuesto.

2.1 Búsqueda Léxica y el Algoritmo BM25

La búsqueda léxica, también denominada bag-of-words, representa documentos como conjuntos no ordenados de términos, ignorando el contexto semántico. Su implementación en bases de datos se da a través de la búsqueda de texto completo (Full-Text Search), disponible en PostgreSQL.

El algoritmo BM25 (Robertson & Zaragoza, 2009) es el estándar de este paradigma. Su fórmula de relevancia para un documento D dado una consulta Q es:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (K_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl}}$$

donde $f(q_i, D)$ es la frecuencia del término, $avgdl$ la longitud promedio de documentos, y b, k_1 parámetros de ajuste. Su solidez teórica y eficiencia explican por qué sigue siendo un pilar en arquitecturas híbridas modernas.

2.2 FTS en PostgreSQL

PostgreSQL implementa Full-Text Search (FTS) con índices y operadores especializados, integrables con SQL estándar y transacciones ACID: ideal para combinar metadatos estructurados y campos textuales. En tu contexto (PYME), FTS aporta un baseline léxico estable que se fusiona después con el ranking vectorial.

2.3 Embeddings

Los embeddings son representaciones numéricas densas de fragmentos de información (palabras, frases, imágenes) en un espacio vectorial de alta dimensión. El principio fundamental es que la distancia en este espacio vectorial se corresponde con la similitud semántica en el mundo real. A diferencia de un one-hot encoding, los embeddings capturan relaciones complejas: palabras con significados cercanos se ubican próximas en ese espacio.

Este "espacio semántico" permite realizar operaciones matemáticas con el significado. La similitud entre dos vectores, que representa la similitud entre dos conceptos, suele calcularse con la similitud coseno (Manning et al., 2008):

$$\sin(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Un valor cercano a 1 indica una alta similitud semántica, mientras que un valor cercano a 0 indica disimilitud, permitiendo comparar textos o imágenes independientemente de las palabras clave que contengan. Los modelos modernos como multilingual-e5-small o gte-multilingual-base son codificadores de texto especializados en producir embeddings de alta calidad para tareas de recuperación.

2.4 RRNs y CNNs

Antes de los Transformers, el PLN se apoyaba en:

- **Redes Neuronales Recurrentes (RNNs):** procesan secuencias palabra por palabra, manteniendo un estado interno que captura dependencias temporales. Son útiles en lenguaje, pero sufren de limitaciones de memoria a largo plazo.
- **Redes Neuronales Convolucionales (CNNs):** originalmente diseñadas para visión por computadora, aplican filtros convolucionales que capturan patrones locales. Se usaron en lenguaje para detectar n-grams o frases clave.

Ambos enfoques fueron superados en flexibilidad y paralelización por la arquitectura Transformer (Vaswani et al., 2017).

2.5 La Arquitectura Transformer y el Mecanismo de Atención

La verdadera revolución en el PLN llegó con la arquitectura Transformer, introducida por Vaswani et al. (2017). A diferencia de las RNNs, que procesan texto de forma secuencial, los Transformers procesan secuencias enteras de una vez. Su innovación clave es el mecanismo de auto-atención (self-attention), que permite al modelo ponderar la importancia de diferentes palabras en una secuencia al codificar una palabra específica.

En la práctica, esto significa que el modelo puede capturar dependencias a larga distancia y entender el contexto de una palabra basándose en toda la oración, no solo en las palabras adyacentes. Por ejemplo, en la frase "El banco emitió un comunicado sobre las tasas del banco de arena", el mecanismo de atención puede diferenciar los dos significados de "banco".

Esta capacidad para generar representaciones vectoriales ricas en contexto es lo que hace que modelos como BERT (Devlin et al., 2019) y sus sucesores sean tan potentes para tareas de búsqueda semántica.

2.6 Modelos de Lenguaje Grandes (LLMs)

Los LLMs (Large Language Models) son modelos neuronales de gran escala (con miles de millones de parámetros), entrenados en vastos corpus de texto. Estos modelos son capaces de comprender y generar lenguaje natural con alta coherencia (Devlin et al., 2019; Lewis et al., 2020). Por ejemplo: GPT, BERT, SBERT, E5. Si bien, la capacidad de estos modelos está en constante mejora, estos tienen varias limitaciones, como lo son: conocimiento estático y tendencia a generar “alucinaciones” (información plausible pero incorrecta).

2.7 Multimodalidad y CLIP

El modelo CLIP (Radford et al., 2021) entrena dos codificadores (imagen y texto) en un mismo espacio vectorial mediante aprendizaje contrastivo. Con esto es posible buscar imágenes a partir de texto y viceversa, habilitando tareas de clasificación zero-shot.

CLIP (OpenAI), aprende un espacio conjunto texto-imagen mediante aprendizaje contrastivo a gran escala; es “estándar” para **zero-shot** y recuperación multimodal. **Limitación conocida:** el encoder de texto original opera con ventanas cortas (**~77 tokens**), lo que restringe las descripciones muy largas.

Jina CLIP v2 (Jina AI), es un modelo multimodal multilingüe reciente con **ventanas largas (≈ 8 K tokens para texto)** y "Matryoshka embeddings" (permiten reducir dimensionalidad sin reentrenar). En escenarios de producto (búsqueda catálogo, activos visuales), reduce complejidad al servir **texto↔texto** y **texto↔imagen** con un solo modelo.

Clip-ViT-B-32-multilingual-v1, desarrollado por Sentence-Transformers, es una versión multilingüe del CLIP-ViT-B/32 y mapea texto (en más de **50 idiomas**) e imagenes a un espacio vectorial compartido. Para lograr esta capacidad multilingüe, utiliza el método de **Multilingual Knowledge Distillation**: entrena un modelo DistilBERT multilingüe (como estudiante) para alinearse con el espacio de embeddings del encoder de imágenes original de CLIP.

2.8 Modelos de Embeddings de Texto

multilingual-e5-small, desarrollado por Microsoft, este modelo genera embeddings de alta calidad para tareas de recuperación de información en más de 100 idiomas. Su ventaja principal es la eficiencia, con un tamaño reducido (384 dimensiones), lo que lo hace ideal para entornos con recursos limitados.

gte-multilingual-base (General Text Embeddings), perteneciente a la Alibaba Group, este modelo se optimiza para la representación de texto en más de 70 idiomas y destaca por su capacidad para manejar contextos largos de hasta 8192 tokens.

Jina Embeddings v3, es un modelo de embeddings multilingüe de texto con **570 millones de parámetros** y capacidad para procesar **secuencias de hasta 8 192 tokens** gracias a la incorporación de **Rotary Position Embeddings (RoPE)** . Además, integra **adaptadores LoRA específicos por tarea** (como *retrieval.query*, *classification*, *text-matching*, etc.), que suman menos del 3 % de los parámetros y permiten ajustar el modelado según el objetivo, sin incrementar significativamente la carga computacional.

2.9 Bases de Datos Vectoriales y la Indexación ANN con HNSW

La búsqueda semántica a escala presenta un desafío computacional: comparar un vector de consulta con millones de vectores en una base de datos (búsqueda exhaustiva o de

fuerza bruta) es inviable en tiempo real. La solución a este problema son los algoritmos de **Búsqueda de Vecinos Próximos Aproximados (Approximate Nearest Neighbors, ANN)**.

En lugar de garantizar encontrar el vecino más cercano, los algoritmos ANN encuentran candidatos muy cercanos con una alta probabilidad, sacrificando una fracción de precisión a cambio de una mejora drástica en la velocidad. Esto ha dado lugar al concepto de base de datos vectorial, sistemas optimizados para almacenar, indexar y consultar embeddings de alta dimensión.

El algoritmo **HNSW (Hierarchical Navigable Small World)** es uno de los métodos ANN más eficientes y populares. Organiza los vectores en un grafo jerárquico multinivel. En los niveles superiores, los enlaces conectan nodos distantes (permitiendo saltos largos a través del espacio vectorial), mientras que en los niveles inferiores, los enlaces son cortos y conectan vecinos cercanos. La búsqueda comienza en el nivel superior, navegando rápidamente hacia la región de interés, y luego desciende a los niveles más detallados para refinar la búsqueda (Malkov & Yashunin, 2018).

Extensiones como pgvector para PostgreSQL integran esta capacidad directamente en una base de datos relacional, permitiendo unificar datos estructurados y vectoriales en un solo sistema.

2.10 Búsqueda Híbrida y RRF

Para combinar búsqueda léxica y vectorial se usa Reciprocal Rank Fusion (RRF) (Cormack et al., 2009), que fusiona listas de resultados según su posición en cada ranking:

RRFd=r∈R1k+rankr(d)

Esto aprovecha la precisión léxica y la flexibilidad semántica.

2.11 Generación Aumentada por Recuperación (RAG)

La arquitectura RAG (Lewis et al., 2020) fue diseñada para mitigar las limitaciones inherentes de los LLMs, como el conocimiento estático y la tendencia a generar "alucinaciones". RAG combina un Retriever (recuperador) con un Generator (generador, un LLM) en un flujo sinérgico:

- **Recuperación (Retrieval):** Ante una consulta del usuario, el Retriever (en este caso, el motor de búsqueda híbrida) busca en una base de conocimientos (los datos indexados) y extrae los fragmentos de información más relevantes.
- **Aumentación (Augmentation):** Los fragmentos recuperados se inyectan como contexto adicional en el prompt que se enviará al LLM, junto con la pregunta original del usuario.
- **Generación (Generation):** El LLM recibe este prompt "aumentado" y genera una respuesta en lenguaje natural, con la instrucción de basarse exclusivamente en el contexto proporcionado.

Este enfoque mejora drásticamente la factualidad y la trazabilidad de las respuestas, ya que estas quedan "ancladas" a los datos reales de la organización. Frameworks como LangChain, se especializan en orquestar este flujo de forma modular y eficiente.

2.12 Métricas de evaluación

Estas métricas evalúan la estructura y coherencia del espacio de *embeddings* sin necesidad de consultas de prueba.

2.12.1 1-NN Accuracy (Leave-One-Out)

Mide la calidad de la representación vectorial. Para cada elemento, se busca a su vecino más cercano (1-NN) en el espacio vectorial. Si el vecino más cercano comparte la misma etiqueta (ej. categoría de producto), se considera un acierto. Una alta precisión indica que los embeddings agrupan semánticamente los elementos de manera efectiva (Cover & Hart, 1967).

- **Silhouette Score:** Cuantifica qué tan bien definidos están los clústeres de datos en el espacio vectorial. Un puntaje alto indica que los elementos están muy cerca de otros en su mismo clúster y lejos de los elementos de clústeres vecinos, lo que sugiere una buena separación semántica (Rousseeuw, 1987).
- **Clustering Purity (ARI & NMI)**e utilizan el Índice de Rand Ajustado (ARI) y la Información Mutua Normalizada (NMI) para medir la pureza de los clústeres formados por los embeddings en comparación con las categorías de producto existentes. El ARI corrige por azar la similitud entre dos particiones (Hubert & Arabie, 1985), mientras

que el NMI mide la dependencia estadística entre los clústeres y las etiquetas reales. Valores cercanos a 1 en ambos indican una alta pureza.

2.12.2 Multimodal Alignment

Estas métricas se centran en evaluar la capacidad del sistema para conectar texto e imágenes de manera coherente.

- **Recall@K (Cross-Modal Retrieval):** Mide la capacidad del sistema para recuperar la imagen correcta a partir de una consulta de texto (y viceversa) dentro de los K primeros resultados. Un alto Recall@K es fundamental para demostrar que el espacio semántico unificado es coherente entre ambas modalidades.
- **Mean Reciprocal Rank (MRR):** Evalúa la posición del primer resultado correcto en una tarea de recuperación cruzada. Es especialmente útil porque da más peso a los sistemas que devuelven el resultado correcto en las primeras posiciones, lo cual es crucial para la experiencia del usuario (Craswell, 2009).
- **Positive vs. Negative Pair Separation:** Mide la distancia en el espacio vectorial entre pares "positivos" (una imagen y su descripción correcta) y pares "negativos" (una imagen y una descripción incorrecta). Una separación clara y amplia es indicativa de un modelo de *embedding* robusto que ha aprendido a distinguir eficazmente las relaciones semánticas.

2.12.3 Hybrid Search Quality

Estas métricas cuantifican el valor añadido de combinar la búsqueda léxica y la semántica.

- **Hybrid Recall:** Mide el porcentaje de resultados relevantes recuperados por el sistema híbrido en comparación con los sistemas léxico y vectorial por separado. El objetivo es demostrar que la fusión (RRF) logra una cobertura de resultados relevantes superior a la de sus componentes individuales.
- **Filter-Separation Score:** Esta métrica evalúa la capacidad del sistema para distinguir entre resultados que coinciden con un filtro explícito (ej. "hamburguesas") y aquellos

que son solo semánticamente similares pero no pertenecen a la categoría. Una buena separación indica que el sistema puede manejar consultas que combinan filtros estructurados con intención semántica.

Capítulo 3. Metodología

La presente investigación se enmarca en la **Investigación en Ciencia del Diseño (Design Science Research, DSR)**, un enfoque ampliamente utilizado en proyectos donde el objetivo principal es la construcción y validación de artefactos tecnológicos (Hevner et al., 2004). Este marco metodológico resulta pertinente, dado que la tesis se centra en concebir, implementar y evaluar un motor de búsqueda híbrido y multimodal orientado a PYMEs.

En particular, se adopta la estructura propuesta por **Peffers et al. (2007)**, que organiza el proceso en seis fases, mencionadas a continuación.

3.1 Fase 1 - Identificación del problema y motivación

Se identificó la brecha tecnológica entre grandes corporaciones, que cuentan con motores de búsqueda semánticos y multimodales propietarios, y las PYMEs, que dependen de motores tradicionales basados en coincidencia léxica. El problema radica en la falta de arquitecturas accesibles que integren de forma unificada recuperación híbrida, multimodalidad y generación de consultas estructuradas (RAG-to-SQL).

3.2 Fase 2 - Definición de los Objetivos de la Solución

El objetivo general es **diseñar y validar una arquitectura tecnológica** basada en tecnologías open-source, que combinen recuperación léxica y semántica (texto e imagen), integrando datos estructurados y representaciones vectoriales, con capacidad de interacción mediante RAG, para emular las capacidades de motores de búsqueda empresariales a un costo accesible para PYMEs.

3.3 Fase 3 - Diseño y Desarrollo del Artefacto

Se adoptó un enfoque **modular e incremental**, que asegura portabilidad tecnológica, validación temprana y viabilidad en entornos de bajo costo. Aunque en el prototipado se emplearon GPUs para acelerar procesos intensivos, la solución final es **agnóstica** a la infraestructura, pudiendo ejecutarse en entornos sin hardware especializado.

Con base en estos lineamientos, la arquitectura se compone de los siguientes módulos:

- **Pipeline de ingesta reproducible** (S3 → Parquet → PostgreSQL).
- **Embeddings multimodales** de texto (E5, JE3, GTE) e imagen (CLIP).
- **Indexación vectorial** con HNSW sobre pgvector.
- **Motor híbrido de búsqueda** (BM25 + vectores, con fusión mediante RRF).
- **Componente RAG** con LangChain y LangGraph para consultas en lenguaje natural y generación de SQL.

3.4 Fase 4 - Demostración

Una vez construido, el artefacto se utiliza para resolver una instancia del problema. En este proyecto, la demostración consiste en la ejecución del prototipo final sobre el dataset de Glovo (FooDI-ML), que incluye información estructurada (pedidos, precios, restaurantes) y no estructurada (descripciones, imágenes).

La demostración consistió en mostrar:

- Recuperación híbrida en consultas textuales.
- Recuperación multimodal texto-imagen.
- Generación de respuestas en lenguaje natural mediante RAG.
- Traducción de consultas en lenguaje natural a SQL ejecutable.

3.5 Fase 5 - Evaluación

La evaluación del artefacto se diseñó con un enfoque multifacético, considerando no solo la precisión de la recuperación, sino también la calidad del ranking, la coherencia multimodal y la efectividad de la búsqueda híbrida. Para ello, se utilizaron métricas específicas agrupadas en tres dimensiones principales: **Relevance & Ranking Quality (Proxy-Based), Multimodal Alignment** & **Hybrid Search Quality (Score@K, PWR, POC, alignment, hybrid gain).**

3.6 Fase 6 - Comunicación

La principal vía de comunicación es el presente **documento de tesis**, complementado por el **repositorio de código fuente público en GitHub: linaB-R/vector-hybrid-search/**, que asegura la transparencia y reproducibilidad de la investigación, aportando la "arquitectura de referencia" como una de las contribuciones clave del trabajo.

3.7 Cronograma

Tabla 1 Cronograma

Mes	Actividad principal	Actividades clave	Herramientas / Técnicas principales
1	Revisión sistemática y definición de arquitectura	Revisión de literatura (DSR, búsqueda híbrida, RAG), definición de objetivos, diseño de arquitectura modular (pipeline, embeddings, motor híbrido, RAG).	Análisis bibliográfico, PostgreSQL, pgvector, diseño conceptual, diagramación de arquitectura
2	Ingesta de datos y generación de embeddings	Implementación de pipeline reproducible (S3 → Parquet → PostgreSQL), limpieza y normalización, generación de embeddings de texto (E5, JE3,GTE) e imagen (CLIP).	Python, PyTorch, HuggingFace Transformers, CLIP, pgvector, CUDA (Colab en prototipado)
3	Desarrollo del motor híbrido y componente RAG	Construcción del índice HNSW en pgvector, integración de BM25 + vectores con RRF, implementación del componente RAG-to-SQL con LangChain y LangGraph.	PostgreSQL, pgvector (HNSW), BM25, RRF, LangChain, LangGraph, desarrollo en Python
4	Demostración y evaluación inicial	Aplicación del sistema al caso de Glovo, pruebas funcionales de búsqueda híbrida y multimodal, medición de métricas (Score@K, PWR, POC, alignment, gain).	Relevance & Ranking Quality, Multimodal Alignment, Hybrid Search Quality, Python, análisis comparativo
5	Optimización, documentación y comunicación de resultados	Ajuste de parámetros (índices, embeddings, fusión), benchmarking, documentación técnica y académica, preparación del informe final y conclusiones.	Benchmarking, tuning HNSW, documentación, redacción académica, GitHub, presentación de resultados

Capítulo 4. Implementación

El desarrollo se llevó a cabo en Python, utilizando un ecosistema de librerías de código abierto. El stack tecnológico principal incluye **PostgreSQL** con la extensión **pgvector** como base de datos unificada, **PyTorch** y **Transformers** de Hugging Face para el modelado semántico, y **LangChain** para la orquestación del flujo de Generación Aumentada por Recuperación (RAG).

A continuación, se presenta el proceso del prototipo del sistema de búsqueda híbrida desarrollado, ilustrado a través de un diagrama.

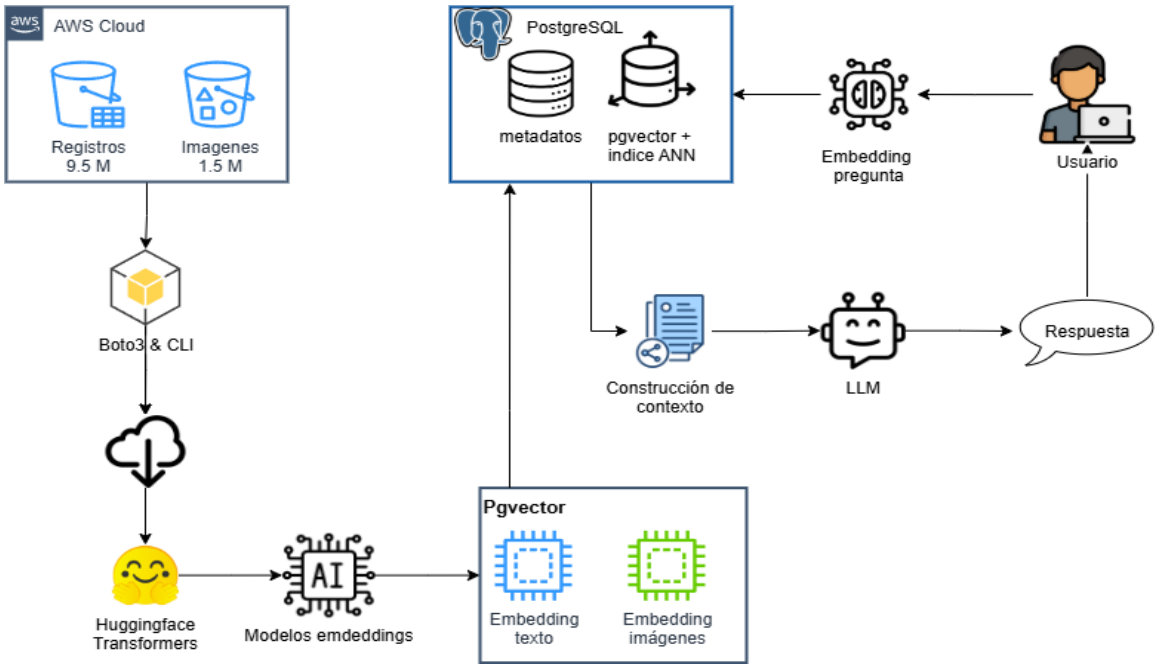

Ilustración 1. Diagrama de flujo del proyecto

4.1 Arquitectura General del Sistema

La arquitectura del sistema está diseñada como un pipeline modular que transforma datos heterogéneos en respuestas inteligentes y trazables. El flujo de datos de extremo a extremo sigue la secuencia:

- **Ingesta:** Los datos se extraen de una fuente de almacenamiento de objetos (AWS S3), se transforman a un formato optimizado (Parquet) y se cargan en la base de datos PostgreSQL.

- **Indexación:** Se generan representaciones vectoriales (embeddings) para el texto y las imágenes, que se almacenan e indexan en pgvector junto a un índice léxico tradicional.
- **Segmentación del conocimiento (*chunking*):** Se emplea un chunking de ventana fija con solapamiento ligero (10–15%), generando fragmentos de 300–400 tokens a partir de product_name + product_description y conservando como metadatos store_name, country_code, city_code y collection_section. Este es el método estándar más simple: es reproducible, funciona bien tanto en CPU como GPU, y ofrece una relación señal–ruido adecuado para el tamaño muestral utilizado en este trabajo. En esta fase se prioriza la simplicidad operativa y la baja latencia porque la muestra no es lo suficientemente grande como para justificar el costo de ingeniería adicional.
- **Recuperación Híbrida:** Ante una consulta, el sistema recupera candidatos tanto del índice léxico (BM25) como del vectorial (HNSW), y fusiona los resultados (RRF) para maximizar la relevancia.
- **Generación Aumentada:** Los candidatos recuperados sirven como contexto para un Modelo de Lenguaje Grande (LLM) que genera una respuesta en lenguaje natural.

4.2 Pipeline de Ingesta y Preparación de Datos

4.2.1 Extracción de Datos desde la Fuente (AWS S3)

El desarrollo y la validación del sistema se llevaron a cabo sobre el conjunto de datos público FooDI-ML de Glovo, una colección de 9.5 millones de registros alojada en AWS S3. Para la construcción del prototipo, se extrajo mediante la librería **boto3** una muestra operativa de **15,000** registros. La selección de esta muestra se realizó tras un proceso de filtrado, donde; se seleccionaron únicamente los registros de países de habla hispana y registros sin datos faltantes en las columnas de interés e imágenes (product_name, collection_section, product_description).

4.2.2 Transformación y Almacenamiento Intermedio

Los datos textuales, inicialmente en formato CSV, fueron procesados y convertidos al formato **Apache Parquet**. Esta elección se justifica por la eficiencia del almacenamiento columnar, la cual optimiza la compresión y acelera significativamente la lectura y pre-procesamiento de datos.

4.2.3 Carga y Estructuración en la Base de Datos

Los datos procesados se cargaron en una instancia de **PostgreSQL** (gestionada a través de Supabase). Previamente, se habilitó la extensión **pgvector** en la base de datos y se diseñó un esquema relacional que incluye columnas de tipo vector para almacenar los futuros **embeddings** de texto e imágenes, junto a los datos estructurados del catálogo.

4.3 Generación y Almacenamiento de Embeddings

4.3.1 Selección de Modelos de Representación

La selección de los modelos de embedding se realizó a través de un proceso de evaluación comparativa para determinar la opción óptima para cada modalidad del proyecto, priorizando el balance entre la calidad de la recuperación y la eficiencia computacional.

Modelos de Texto Multilingüe:

Se compararon los modelos **multilingual-e5-small** y **gte-multilingual-base**, ambos con un sólido rendimiento reportado en el benchmark **MTEB** para tareas de recuperación multilingüe. Además, se incorporó el modelo **jina-embeddings-v3** (JE3), un modelo avanzado multilingüe de alto rendimiento que supera a modelos como los propietarios de OpenAI y Cohere, y que incluso supera a **multilingual-e5-large-instruct** en todas las tareas multilingües del benchmark **MTEB**, manteniendo además una mayor eficiencia y soporte para entradas de texto muy largas (hasta 8192 tokens) gracias a técnicas como *late chunking* y LoRA-adapters.

Modelos Multimodales (Texto e Imagen)

Se evaluaron los modelos de embeddings para imágenes **openai/clip-vit-base-patch32** y **sentence-transformers/clip-ViT-B-32-multilingual-v1**, ambos mapean texto e imágenes a un espacio semántico común. El último es una versión multilingüe del CLIP-ViT-B/32 que permite procesar texto en más de 50 idiomas alineado con el encoder de imágenes.

También se exploró el modelo multimodal **jina-clip-v2**, que sobresale tanto en recuperación **texto → texto** como en **texto ↔ imagen**, sirviendo de puente eficiente entre modalidades sin requerir modelos separados. **Por costos de computación no se utilizó para la evaluación final.**

Las imágenes utilizadas para la evaluación fueron procesadas directamente desde su origen en buckets de AWS S3.

El procesamiento multimodal se realiza en dos pasos claramente diferenciados:

1. **Imagen → Texto**: se genera embedding de la imagen (con clip-ViT), y se calcula su similitud con embeddings de texto para recuperar descripciones o información textual relevante.

2. **Texto → Imagen**: se genera embedding del texto (incluido texto multilingüe si aplica), y se compara con embeddings de imágenes para recuperar las imágenes más pertinentes.

4.3.2 Generación de Vectores (Batch Processing)

La vectorización de los **15.000 registros** se ejecutó en modo **offline** mediante **cómputo por lotes en GPU** (Google Colab con acelerador CUDA), fijando un **batch size de 512** para maximizar el rendimiento sin exceder memoria de dispositivo.

El flujo implementa: (i) lectura por lotes desde PostgreSQL, (ii) **normalización ligera de texto** (minúsculas, NFKC, limpieza de URLs y símbolos), (iii) inferencia batcheada del modelo (GPU) y (iv) **actualización masiva** de la columna vectorial mediante `UPDATE ... FROM (VALUES ...)` con `execute_values`, grabando cada embedding en el tipo **pgvector**. Este enfoque amortiza I/O y reduce viajes a base de datos, a la vez que aprovecha el paralelismo de CUDA para acelerar la inferencia por lotes.

Colab permite habilitar GPU/T4 y soporte CUDA para PyTorch; los lotes grandes tienden a **mejorar throughput** (más items/seg), aunque con posibles costos de latencia y uso de memoria. Para cargas intensivas, técnicas como **mixed precision (AMP)** ayudan a reducir memoria y aumentar velocidad en GPUs con Tensor Cores.

El mismo procedimiento se aplicó **de forma análoga** a los demás modelos de Embeddings tanto para texto como para imgenes.

4.4 Motor de Recuperación Híbrida (Recall)

El núcleo del sistema es un motor de recuperación que combina lo mejor de los mundos léxico y semántico para maximizar la cobertura (recall) de resultados relevantes.

4.4.1 Implementación de la Búsqueda Léxica (BM25)

Se utilizó la funcionalidad de **Full-Text Search (FTS)** nativa de PostgreSQL, configurando un índice **tsvector** sobre las columnas textuales. Este índice permite realizar búsquedas léxicas eficientes que, aunque no capturan la semántica, son robustas ante términos raros, códigos de producto y errores tipográficos menores. El algoritmo de ranking subyacente es una implementación de **BM25**.

4.4.2 Implementación de la Búsqueda Vectorial (ANN-HNSW)

Para la búsqueda semántica, se construyó un índice **HNSW (Hierarchical Navigable Small Worlds)** sobre cada columna de tipo vector utilizando pgvector. Los hiperparámetros del índice (m, ef_construction, ef_search) se ajustaron experimentalmente para encontrar un balance óptimo entre la precisión de la recuperación y la latencia de la consulta, de acuerdo con las recomendaciones de la documentación de pgvector y la literatura sobre ANN.

4.4.3 Fusión de Resultados: Sin fusión obligatoria

El sistema opera **solo con el índice primario** (**vectorial e5 u otros embeddings + filtros estructurados**) y **ordenar por su puntuación nativa** (coseno/HNSW).

4.5 Orquestación y Generación de Respuestas (RAG)

4.5.1 Procesamiento de la Consulta de Entrada (Query Pipeline)

Cuando un usuario introduce una consulta, esta pasa por un pipeline de optimización antes de la recuperación:

- **Generación de Embedding:** La consulta se convierte en un vector utilizando el mismo modelo E5 u otros vectores disponibles que se usó para la indexación.
- **Extracción de Filtros:** Se extraen de la consulta posibles filtros estructurados que se aplicarán como cláusulas WHERE u otros filtros en SQL.

4.5.2 Implementación del Flujo RAG con LangChain

Se utilizó la librería **LangChain** para orquestar el flujo RAG. El motor de búsqueda híbrida actúa como el "Recuperador" (**Retriever**). El flujo sigue el patrón canónico de RAG (Lewis et al., 2020): se recuperan los top-k documentos relevantes y se inyectan en una plantilla de prompt. Durante la indexación, los documentos se dividen en **chunks de ventana fija** (tamaño constante con solapamiento ligero), lo que simplifica la orquestación y mantiene la consistencia del contexto en la generación.

4.5.3 Generación de Respuestas con Trazabilidad

El *prompt* final, que contiene las instrucciones, la pregunta del usuario y el contexto recuperado, se envía a un LLM. La instrucción principal fuerza al modelo a basar su respuesta **exclusivamente** en la evidencia proporcionada y a **citar** sus fuentes (trazabilidad mínima hacia los fragmentos recuperados).

4.5.4 Generación de Respuestas con Trazabilidad

- **Regla de evidencia en el *prompt***: "Responde solo con la información del contexto; si falta evidencia, di *'No se encontró evidencia suficiente en la base'* y detente."
- **Capado de contexto**: **k = 6–8** y **máx. 1200 tokens** de contexto; recorte por oración/párrafo (no palabra a palabra).
- **Trazabilidad mínima en la salida**: listar **2–3 referencias** del contexto usado (*store_name*, *product_name* y *s3_url/id*).
- **Determinismo**: `temperature = 0` y `top_p = 0.1` para respuestas estables.
- **Fallback seguro**: si `k = 0` tras aplicar filtros o no hay evidencia suficiente, emitir el mensaje de ausencia de evidencia y **no** inventar contenido.
- **Idioma**: responder en el idioma detectado (ES por defecto).
- **Segmentación explícita**: mantener **chunks de ventana fija** (con solapamiento ligero) y evitar heurísticas complejas de *chunking* semántico para no aumentar la superficie de preguntas técnicas.
-

4.6 Preparación para la Evaluación

Para garantizar una validación rigurosa y cuantitativa del artefacto, se diseñó un entorno de evaluación que trasciende las métricas tradicionales de recuperación de información (IR). Este entorno incorpora un conjunto de métricas personalizadas, orientadas a medir el rendimiento desde una perspectiva práctica y centrada en el producto.

Análisis de Confianza del Ranking (Score@K): Para diagnosticar la confianza del sistema en sus resultados, se registró el Score@K. Este indicador corresponde a la puntuación de similitud bruta (coseno o BM25) del ítem en la posición K, permitiendo observar cómo decae la certeza a lo largo de la lista de resultados.

Comparación de Sistemas (Predicted Win Rate - PWR): Para comparar directamente la efectividad del sistema híbrido frente al baseline léxico, se implementó el Predicted Win Rate (PWR). Esta métrica fue utilizada para estimar la probabilidad de que un usuario prefiera

la lista de resultados del sistema propuesto, sirviendo como un indicador clave del éxito relativo.

Calidad del Ordenamiento (Probability of Choice - POC): Para evaluar si los resultados más importantes aparecen primero, se calculó la Probability of Choice (POC). Este indicador mide la probabilidad de que el resultado más relevante para el usuario se encuentre en las primeras posiciones, penalizando a los sistemas que ocultan los mejores ítems en posiciones inferiores.

Medición del Impacto Global (Hybrid Gain): Finalmente, para cuantificar el beneficio neto de la arquitectura, se definió la Ganancia Híbrida (Hybrid Gain). Este indicador agregado sintetiza la mejora en la calidad del sistema fusionado en comparación con el rendimiento de sus componentes por separado, sirviendo como la medida final del éxito.

Capítulo 5. Resultados

5.1 Diseño de la evaluación y datos

La evaluación se llevó a cabo sobre una **muestra operativa de 15.000 registros** extraídos del dataset público **FooDI-ML de Glovo** (AWS S3), filtrando países hispanohablantes y registros completos en las columnas de interés con imagen. Esta muestra se utilizó para **indexación léxica (BM25)** y **vectorial (pgvector/HNSW)**.

El entorno de evaluación consideró, además de métricas IR estándar, indicadores operativos definidos para el prototipo (Score@K, PWR, POC y Hybrid Gain), con el objetivo de medir no solo la recuperación, sino también la **calidad percibida del ranking** y la **ganancia del enfoque híbrido**.

5.2 Relevance & Ranking Quality (Proxy-Based)

Se evaluó la calidad del espacio de embeddings utilizando **collection_section** como proxy de relevancia, midiendo 1-NN (leave-one-out), Silhouette (coseno), ARI/NMI (KMeans) y Label Consistency@K con $K \in \{1,5,10\}$. En este escenario, el modelo e5 mostró el mejor desempeño global:

Tabla 2 Métricas globales por modelo (N, etiquetas únicas, 1-NN, Silhouette, ARI, NMI).

model	1-NN	SILHOUETTE	ARI	NMI	unique_labels
e5	0.189	-0.103622191	0.062921	0.884516	1163
gte	0.1565	-0.121203296	0.047155	0.880068	1163
je3	0.163	-0.135115728	0.045697	0.880003	1163

Según los resultados de la tabla 1 se tiene que:

En la métrica **1-NN (leave-one-out),** *e5* obtiene la mayor exactitud (0,189), por encima de *je3* (0,163) y *gte* (0,1565), lo que indica una **geometría local** del embedding más coherente con el proxy collection_section.

Silhouette: Los tres modelos presentan valores negativos, consistentes con solapamiento entre clases y alta fragmentación del proxy (muchas etiquetas con pocos ítems). e5 es el menos negativo, sugiriendo ligeramente mejor separabilidad.

ARI / NMI: e5 alcanza el mayor ARI (~0,063), mientras que NMI es similar entre modelos (~0,88). Esto sugiere que el alineamiento global con el proxy es comparable, y que la ventaja de e5 proviene principalmente de su estructura local.

Label Consistency@K. *e5* mantiene el mejor porcentaje de coincidencias en $K \in \{1,5,10\}K$ evidenciando mejor ordenamiento en **top-K.**

Tabla 3 Label Consistency@K por modelo.

model	k	label_consistency
e5	1	0.189
gte	1	0.1565
je3	1	0.163
e5	5	0.1144
gte	5	0.1025
je3	5	0.1003
e5	10	0.09005
gte	10	0.0818
je3	10	0.08125

En general, se observa que e5 ofrece una estructura local más alineada con el proxy, útil para vecindarios top-K y como señal de pre-ranking en pipelines híbridos. A continuación, se presentan los resultados y comparativa de los distintos modelos para las métricas del modelo Relevance & Ranking Quality.

- **1-NN Accuracy**

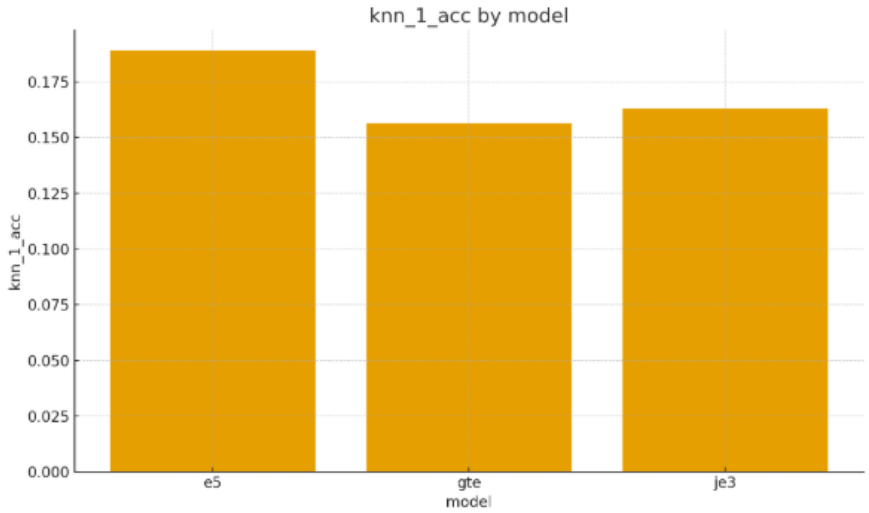

Ilustración 2 1-NN Accuracy

- **Silhouette (coseno):**

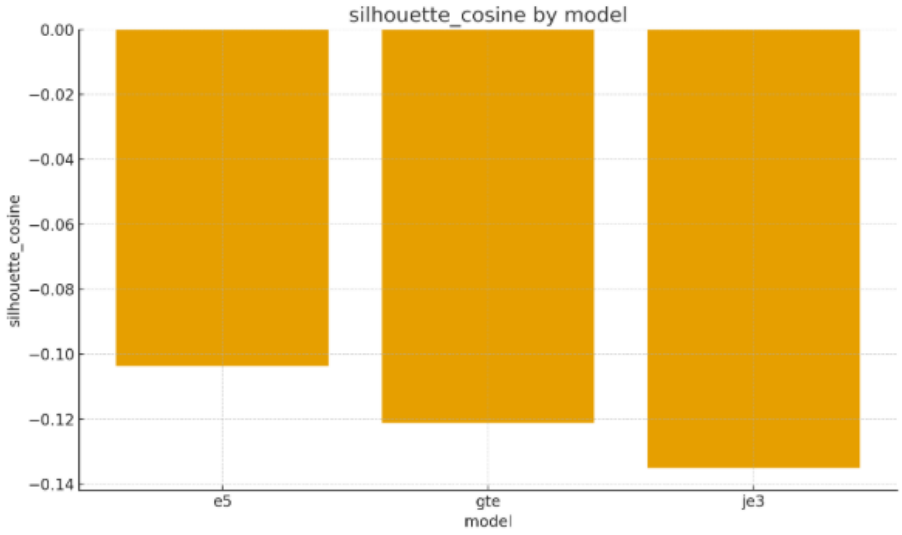

Ilustración 3 Silhouette por modelo

- **ARI**

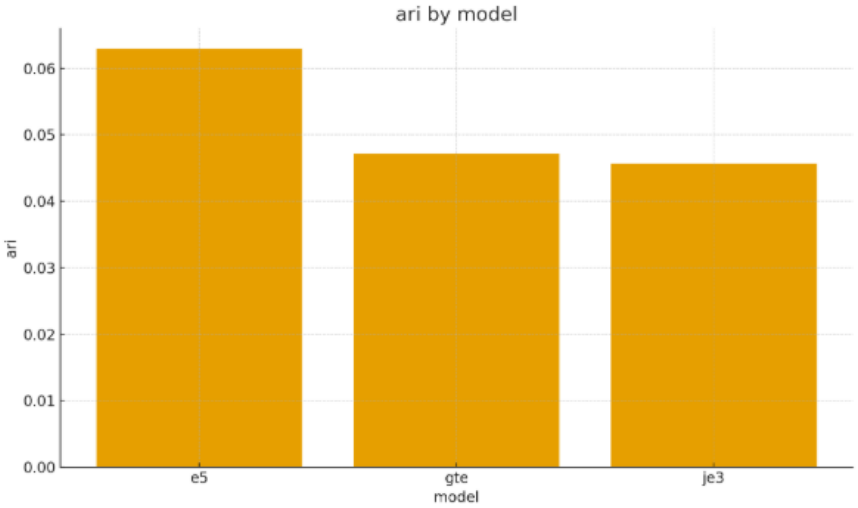

Ilustración 4 ARI por modelo

- **NMI**

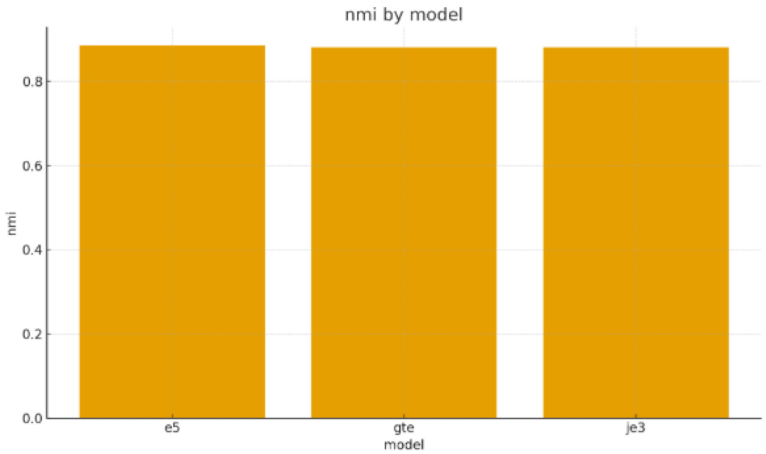

Ilustración 5 NMI por modelo

- **Label Consistency@1/@5/@10**

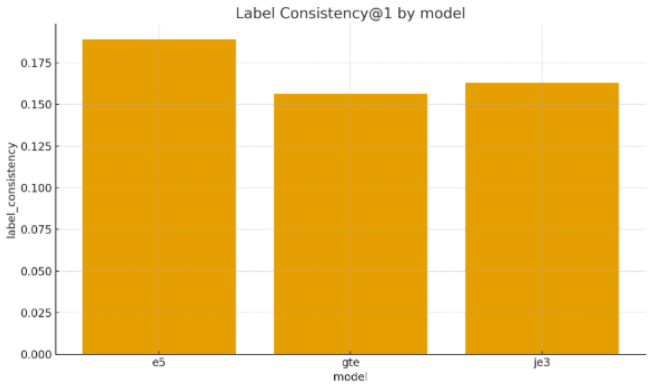

Ilustración 7 Label Consistency@1 por modelo

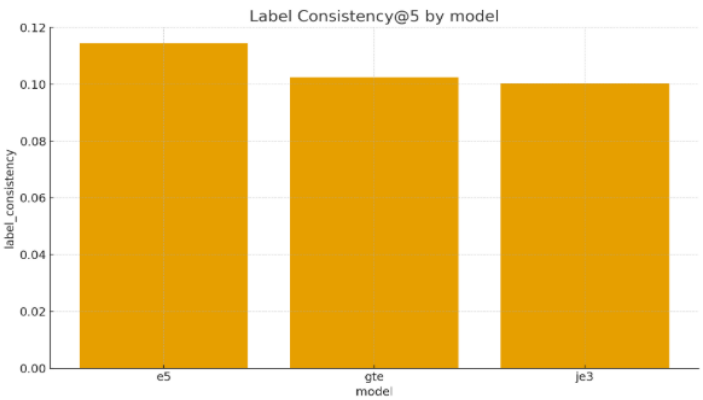

Ilustración 6 Label Consistency@5 por modelo

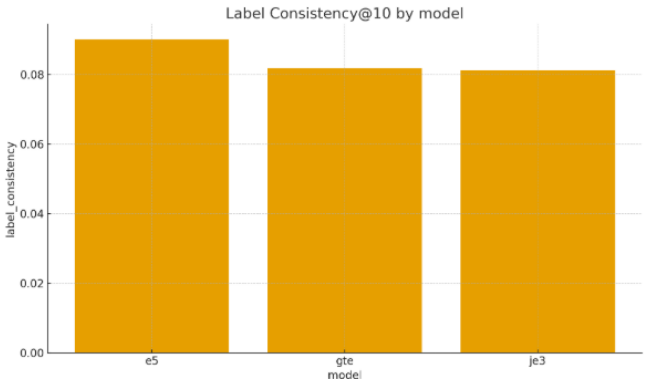

Ilustración 8 Label Consistency@10 por modelo

5.3 Análisis por store_name (proxy alternativo)

Rendimiento global por modelo : gte lidera en 1-NN (0,217) y en Label Consistency@K para K∈{1,5,10}; e5 queda detrás y je3 tercero. En ARI, gte también va primero (0,142), mientras que en NMI la diferencia es mínima (gte ≈0,928, e5 ≈0,926, je3 ≈0,925). Silhouette es levemente negativo para todos (mezcla entre tiendas), siendo je3 el más bajo (−0,101).

Efecto de la **partición por país:** Con muchas etiquetas por país y tamaños desbalanceados, la separabilidad absoluta cae (silhouette≤0 en países grandes), pero emergen bolsillos locales robustos en países medianos/pequeños (CR, DO, PA, GT), donde ARI y 1-NN suben marcadamente y pueden superar 0,30–0,40 de ARI.

Tabla 4 Métricas globales por modelo (proxy = store_name).

model	1-NN_acc	Silhouette	ARI	NMI	LC@1	LC@5	LC@10	unique_labels
e5	0.198	-0.071500547	0.120395	0.926092	0.198	0.1056	0.07255	1310
gte	0.2175	-0.070210844	0.141607	0.927703	0.2175	0.1162	0.08545	1310
je3	0.1835	-0.101479582	0.122086	0.924717	0.1835	0.0985	0.0701	1310

Patrones por país (muestras principales):

ES: diferencias pequeñas; **gte** gana en @K y 1-NN (0,209 vs 0,200 e5), silhouette aún negativo (≈−0,066 a −0,098).

PE y AR: gte domina en @K y ARI; **e5** compite en 1-NN (AR 0,261 e5 vs 0,245 gte; PE 0,236 e5 vs 0,259 gte).

GT/PA/CR: gte o **je3** pueden tomar la punta en **ARI** (p.ej., CR: je3 ≈0,516), mientras **gte** mantiene ventaja en 1-NN y Consistency@K en GT/PA. Signal: **mercados medianos** parecen más "limpios" geométricamente.

Tendencia @K: La consistencia@K decrece al crecer K, pero gte conserva la ventaja relativa a @5 y @10 en ES/PE/AR, lo que sugiere mejor estructura de vecindad ampliada.

Con estos resultados se puede decir que:

1. **Espacio principal:** si se prioriza **recuperación@K** y **1-NN** en nombres de tienda, **gte** es la mejor base para el **primer retrieve**.
2. **Reranking híbrido:** dado que **je3** y **e5** ganan **ARI** en algunos países (p. ej., je3 en CR), un **reranking multi-modelo** (top-M de gte → rerank con e5/je3/LLM-score) puede capturar señales locales.
3. **Filtrado estructural:** con silhouette negativo en países grandes, conviene filtrar por **país** (y si aplica, familia de tiendas) **antes** del re-ranking para reducir mezcla.
4. **Muestreo balanceado:** evaluar métricas **por país con ponderación uniforme** (macro-promedio) además del promedio global, para evitar que ES domine el diagnóstico.

country	model	1-NN_acc	Silhouette	ARI	NMI	LC@1	LC@5	LC@10
ES	e5	0.199623	-0.06303	0.160141	0.923422	0.199623	0.10678	0.072693
ES	gte	0.20904	-0.06594	0.151865	0.923117	0.20904	0.112618	0.078249
ES	je3	0.179849	-0.09844	0.16276	0.922406	0.179849	0.091149	0.065443
PE	e5	0.234568	-0.03441	0.215359	0.926339	0.234568	0.098765	0.060905
PE	gte	0.259259	-0.0176	0.299865	0.93728	0.259259	0.116049	0.074897
PE	je3	0.213992	-0.04108	0.166258	0.921764	0.213992	0.093004	0.063374
AR	e5	0.26087	-0.0195	0.174279	0.897065	0.26087	0.162319	0.114976
AR	gte	0.246377	-0.02105	0.216764	0.901505	0.246377	0.173913	0.133816
AR	je3	0.251208	-0.04276	0.23928	0.904679	0.251208	0.155556	0.118841
EC	e5	0.219101	0.014475	0.313367	0.953817	0.219101	0.068539	0.040449
EC	gte	0.235955	0.018148	0.230506	0.95011	0.235955	0.077528	0.044382
EC	je3	0.174157	-0.00478	0.331296	0.95678	0.174157	0.066292	0.038202
GT	e5	0.291667	0.011004	0.322175	0.901928	0.291667	0.120833	0.066667
GT	gte	0.364583	0.046175	0.339529	0.909273	0.364583	0.135417	0.080208
GT	je3	0.270833	-0.00079	0.274211	0.897726	0.270833	0.110417	0.070833
PA	e5	0.344262	0.046874	0.317779	0.93391	0.344262	0.118033	0.070492
PA	gte	0.295082	0.047233	0.383571	0.941999	0.295082	0.127869	0.072131
PA	je3	0.262295	0.030093	0.436171	0.941178	0.262295	0.12459	0.07377
CR	e5	0.305085	0.068351	0.394717	0.951466	0.305085	0.081356	0.044068
CR	gte	0.322034	0.091623	0.394717	0.9519	0.322034	0.088136	0.045763
CR	je3	0.305085	0.076774	0.514695	0.961405	0.305085	0.081356	0.044068
CL	e5	0.121212	-0.01056	0.103261	0.913222	0.121212	0.054545	0.030303
CL	gte	0.181818	-0.01791	0.238462	0.929109	0.181818	0.054545	0.039394
CL	je3	0.060606	-0.07648	0.09589	0.911459	0.060606	0.024242	0.024242
HN	e5	0.290323	0.038757	0.316804	0.910166	0.290323	0.16129	0.087097
HN	gte	0.290323	0.018734	0.263658	0.898811	0.290323	0.135484	0.080645
HN	je3	0.290323	0.04097	0.199262	0.885749	0.290323	0.148387	0.080645
DO	e5	0.25	0.065788	0.322638	0.950273	0.25	0.06	0.03
DO	gte	0.2	0.061726	0.661319	0.975136	0.2	0.05	0.03
DO	je3	0.2	0.026131	0.661319	0.975136	0.2	0.04	0.03
PR	e5	0.375	0.009628	0.34375	0.878841	0.375	0.125	0.107143
PR	gte	0.375	0.034347	0.34375	0.878841	0.375	0.125	0.107143
PR	je3	0.375	0.016776	0.34375	0.878841	0.375	0.15	0.107143

Tabla 5 Métricas por país y modelo

Comparación de métrica @K global:

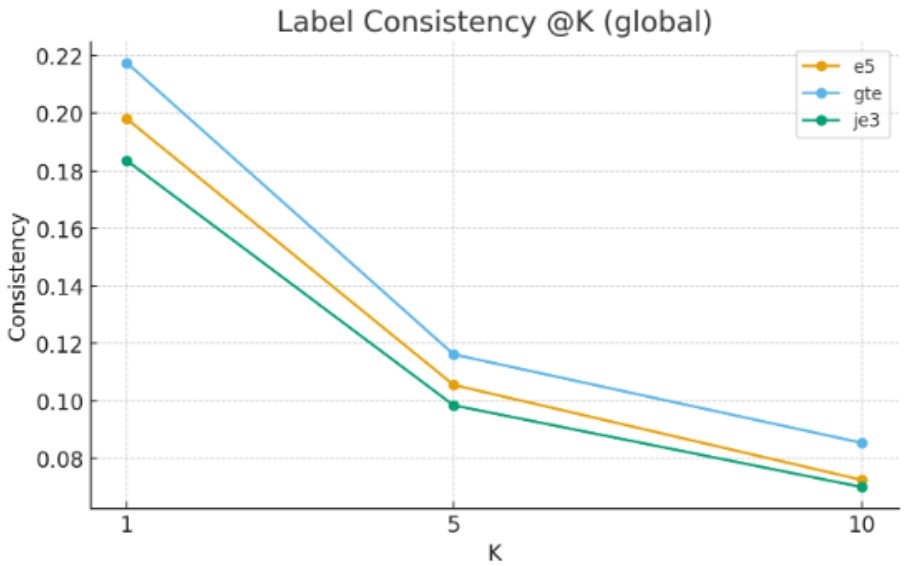

Ilustración 9 Comparación de métrica @K global

5.4 Hybrid Search Quality

Se compararon e5, gte y je3 en un escenario híbrido (vectorial + filtros estructurados), evaluando 8.614 consultas con pool medio ≈792 ítems por filtro. e5 lideró el Recall@K dentro del conjunto filtrado (@1 = 0,080, @5 = 0,194, @10 = 0,258), mientras que gte/je3 quedaron muy próximos entre sí (@10 ≈0,244–0,245). En Filter-Separation, je3/gte presentaron mayores gaps intra vs inter para store_name y collection_section, lo cual es útil como señal de reranking complementaria.

Implicación. En pipelines con filtros por país/ciudad/tienda, el sistema debería recuperar con e5 para maximizar el recall y luego re-rankear incorporando la separación de je3/gte (y, si aplica, puntuación LLM ligera) sobre el top-M.

Tabla 6 Resumen de Recall@K (híbrido)

model	recall@1	recall@5	recall@10	queries_evaluated
e5	0.07990767	0.193857363	0.257670446	8614
gte	0.07565242	0.182960148	0.244041077	8614
je3	0.07383098	0.182960148	0.245391452	8614

Tabla 7 Resumen de Filter-Separation por faceta

model	facet	intra_mean	inter_mean	separation
e5	collection_section	0.894933241	0.852989972	0.041943269
gte	collection_section	0.666226633	0.517150993	0.149075641
je3	collection_section	0.647191949	0.462997252	0.184194697
e5	store_name	0.896647424	0.857499747	0.039147677
gte	store_name	0.681158015	0.520996356	0.160161659
je3	store_name	0.645431935	0.464716625	0.18071531

A continuación, se presenta la comparación de tendencias en las métricas Recall@K visualmente:

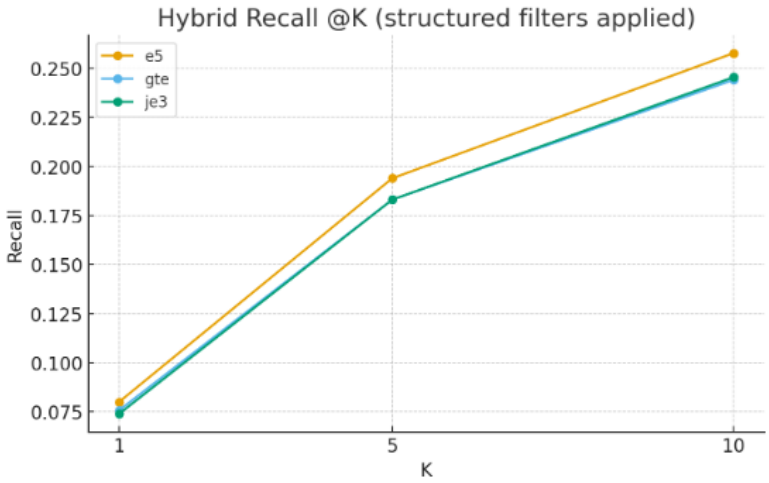

Ilustración 10 Tendencia Recall@K (híbrido)

comparación de métricas Recall@K (híbrido) por modelo:

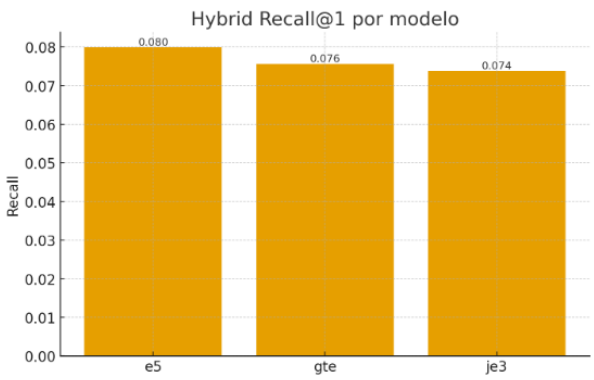

Ilustración 11 Recall@1 (híbrido)

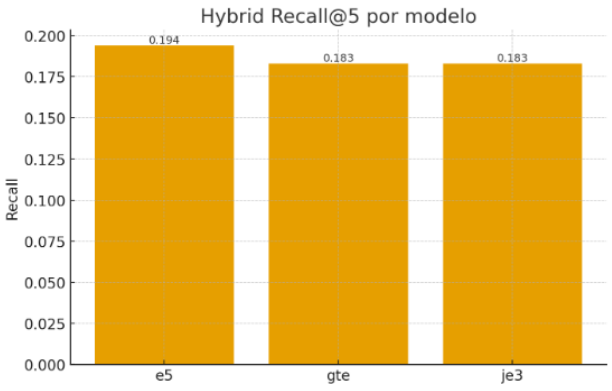

Ilustración 12 Recall@5 (híbrido)

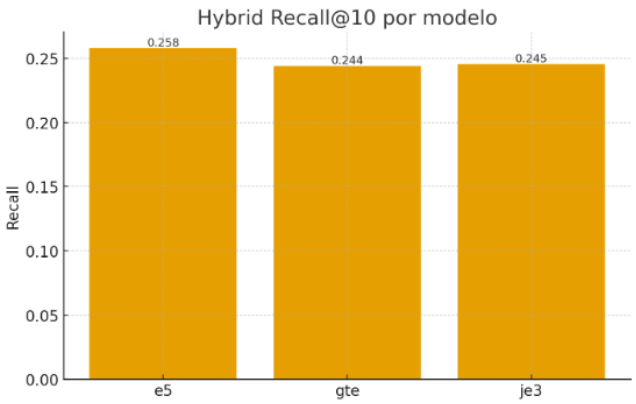

Ilustración 13 Recall@10 (híbrido)

Filter-Separation y promedios intra/inter por faceta:

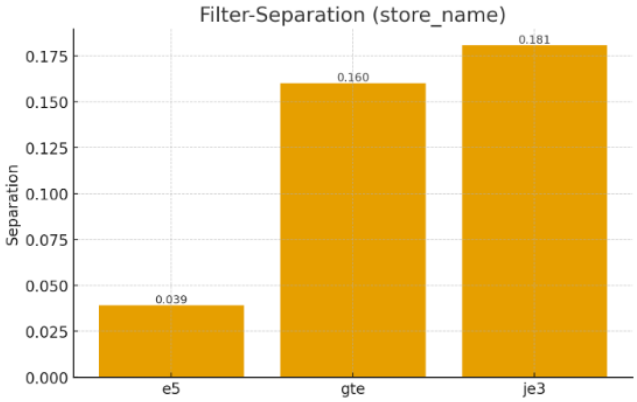

Ilustración 14 Filter separation por store_name (hybrid)

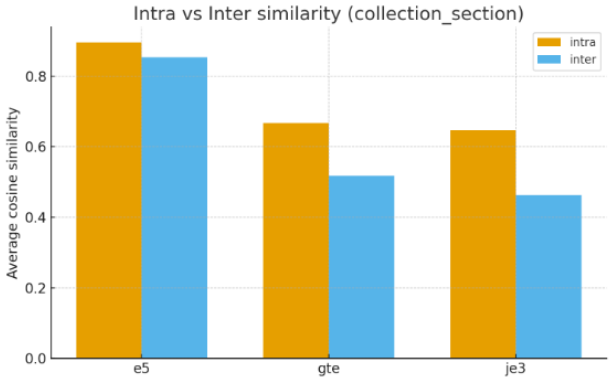

Ilustración 16 Intra vs similarity segun collection_section

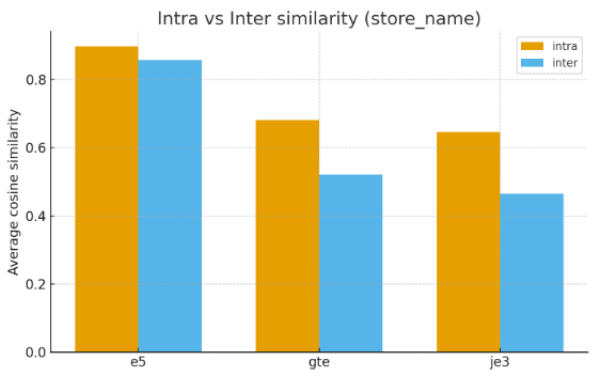

Ilustración 15 Intra vs similarity segun store_name

5.5 Alineación Multimodal (texto ↔ imagen)

Se evaluó la alineación texto↔imagen sobre 15.000 pares producto–descripción (mismo pool de consulta y de galería). Se reportan Recall@K con K∈ $\{1,5,10\}$, MRR y la separación de pares positivos vs. negativos (coseno y euclidiana). Las curvas y comparativas se construyen por dirección de recuperación: texto→imagen (T→I) e imagen→texto (I→T).

Hallazgos cuantitativos

Recall@K (curvas por K + baseline aleatorio): El *recall* crece monótonamente con K en ambas direcciones, ubicándose sistemáticamente por encima del **baseline aleatorio** (K/N):

- **T→I:** R@1 = **0,0185**, R@5 = **0,0485**, R@10 = **0,0705**.
- **I→T:** R@1 = **0,0203**, R@5 = **0,0551**, R@10 = **0,0801**. La **brecha** (I→T − T→I) se **amplía con K**: ≈0,0017 (@1), ≈0,0066 (@5) y ≈0,0096 (@10), lo que indica una ventaja sostenida de I→T a medida que se permiten más candidatos.

MRR (comparación por dirección): I→T presenta MRR mayor (0,0424) frente a T→I (0,0382), consistente con una mejoría en el rango promedio del par correcto cuando la consulta es una imagen.

Separación de pares (positivos vs. negativos)

- **Coseno:** μ_pos ≈ **0,275** vs μ_neg ≈ **0,224** → **Δcos ≈ +0,051**.
- **Euclidiana:** μ_pos ≈ **1,204** vs μ_neg ≈ **1,245** → **Δeuc ≈ +0,041** (distancias menores en positivos).
  En ambos espacios se observa **brecha favorable**: los pares verdaderos quedan, en promedio, más próximos que los aleatorios.

Interpretación de los resultados:

Ventaja direccional I→T. La curva de I→T domina a T→I para todos los K y también en MRR. Esto sugiere que, en este conjunto, las **imágenes** proveen una señal más estable para ubicar su descripción textual que a la inversa (donde el texto puede ser más heterogéneo/ruidoso).

Escala del pool y ruido semántico. Con **15k** candidatos, incluso *recalls* del 1–2% en @1 están **muy por encima del azar**, pero aún dejan **margen de mejora** para hits exactos al tope. El crecimiento con K confirma que existen múltiples candidatos cercanos semánticamente.

Brechas positivas en coseno y euclidiana. Las diferencias **Δcos** y **Δeuc** corroboran la presencia de **alineación contrastiva efectiva** entre pares texto–imagen verdaderos, aunque todavía moderada en magnitud, coherente con la complejidad del dominio.

Implicaciones practices:

Mejorar T→I con texto más discriminativo. Enriquecer descripciones con atributos **concretos** (marca, tamaño, sabor, presentación) y normalizar nombres/taxonomías eleva la precisión de T→I sin perjudicar I→T.

Re-ranking y filtros estructurados. Combinar el puntaje multimodal con **filtros** (tienda, ciudad, categorías agregadas) puede aumentar la precisión práctica en @K.

Entrenamiento contrastivo con "hard negatives". Incorporar negativos cercanos y **fine-tuning** de embeddings en el dominio incrementa la **brecha de separación** y desplaza la curva de Recall@K hacia arriba.

Tabla 8 Recall@K por dirección.

direction	k	recall
texto→imagen	1	0.018533333
texto→imagen	1	0.020266667
texto→imagen	5	0.048466667
texto→imagen	5	0.055133333
texto→imagen	10	0.070533333
texto→imagen	10	0.080133333

Tabla 9 MRR por dirección.

direction	mrr
texto→imagen	0.03815617
imagen →texto	0.04237947

Tabla 10 Separación de pares

pairs	pos_cos_mean	neg_cos_mean	cos_separation	pos_euc_mean	neg_euc_mean	euc_separation
15000	0.275012197	0.2242434	0.050768797	1.203910904	1.245344797	0.041433893

A continuación, se observa la comparación del Recall@K por dirección (con baseline K/N) y **Brecha I→T−T→I**.

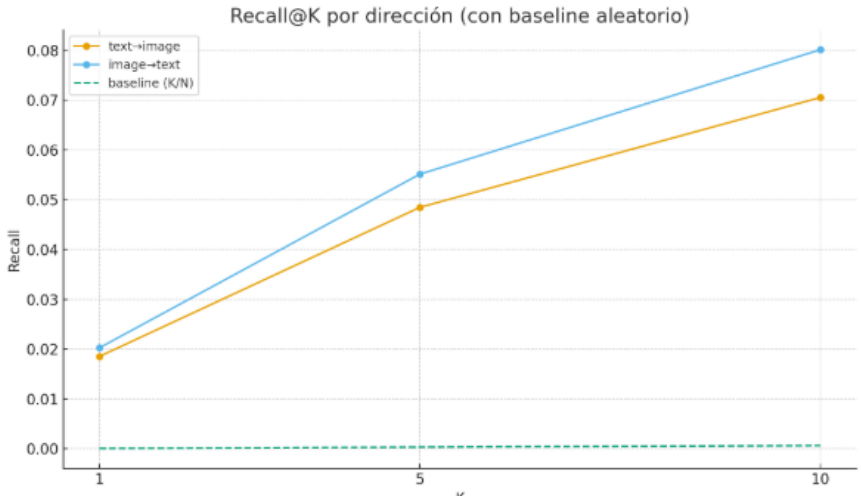

Ilustración 17 Recall @K por dirrección (aleatorio)

Ilustración 18 Brecha Recall (I-T-T-I) por K

Resultados métrica MRR:

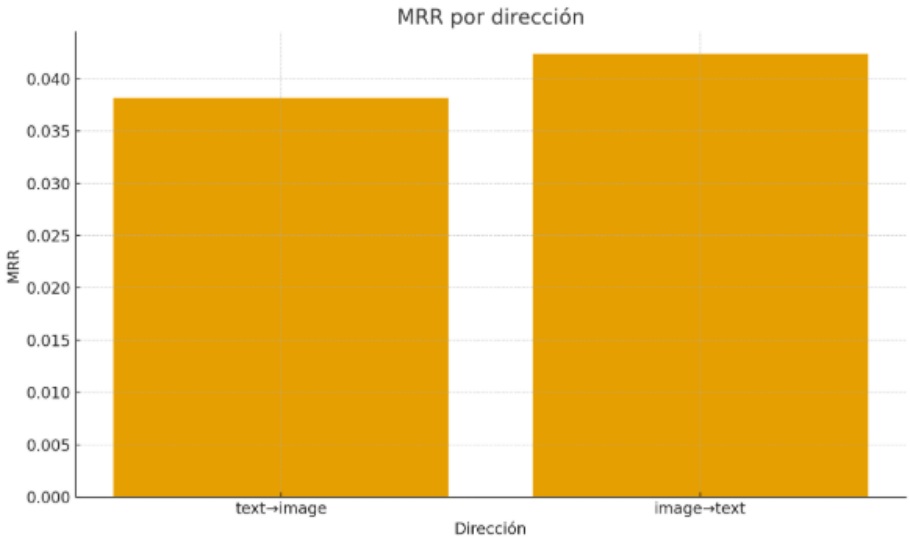

Ilustración 19MRR por dirección

Resultados comparación pares negativos vs positivos:

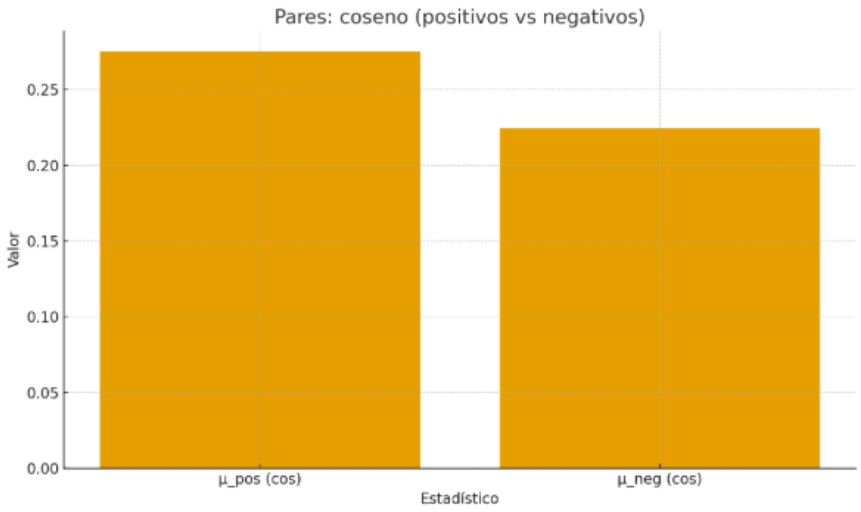

Ilustración 20 Pares positivos vs negativos

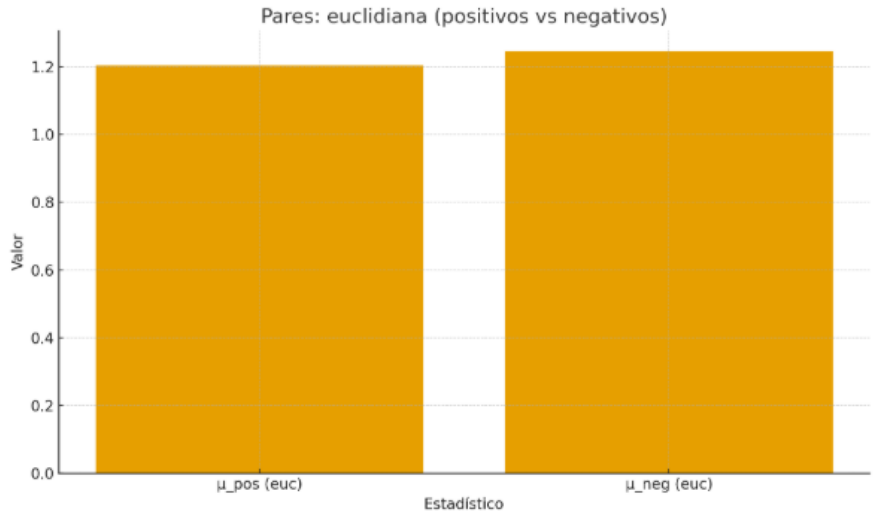

Ilustración 21 Eclidiana - positivos vs negativos

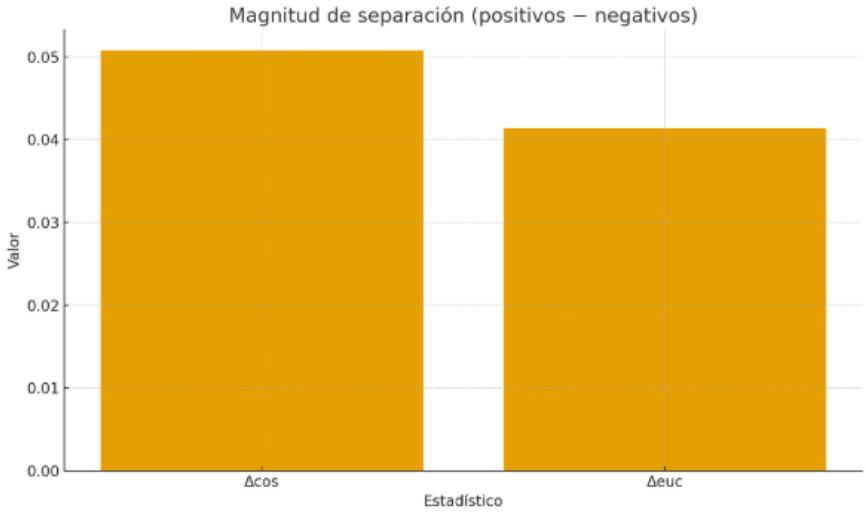

Ilustración 22 Magnitud de separación positivos - negativos

5.6 Discusión de resultados

A grandes rasgos se encontró que, **e5** maximiza el recall@K en **escenarios con filtros estructurados**, lo que lo convierte en una buena base de retrieve. Por otro lado, **gte/je3** aportan señales de separación útiles para reranking y para reforzar la coherencia con facetas como store_name o collection_section.

En cuanto a la **multimodalidad**, la dirección I→T tiene ventaja consistente, atribuible a la variabilidad del texto frente a la estabilidad visual de las imágenes. El **Silhouette negativo** recurrente sugiere mezcla entre clases debida a taxonomías finas o **ruidosas**, lo que respalda normalizaciones y agrupación de etiquetas en etapas de data curation.

Estos hallazgos apoyan el diseño del motor híbrido y su flujo RAG ya descritos, y motivan el uso de indicadores operativos (Score@K, PWR, POC, Hybrid Gain) para capturar el impacto práctico de las decisiones de arquitectura.

5.7 Limitaciones y amenazas a la validez

Limitaciones observadas:

- **Fragmentación de etiquetas** en *collection_section*: muchas clases con pocos ejemplos, lo que reduce 1-NN y Silhouette.
- **Desbalance por país/tienda**: países con más datos pueden dominar promedios globales; se recomienda monitorear **macro-promedios por país**.
- **Calibración de similitudes** entre modelos: escalas diferentes dificultan la fusión directa; se sugiere **normalización** (z-score por lote/tienda o *temperature scaling*) previa al ensamble.

Capítulo 6. Conclusiones

Este trabajo demuestra que, con herramientas abiertas y gratuitas, es posible dotar a PYMEs de un motor de búsqueda híbrido y multimodal (texto–imagen) robusto, construido sobre PostgreSQL + pgvector y técnicas estándar como BM25, HNSW y RRF, y orquestado con LangChain para un flujo RAG-to-SQL. La arquitectura, implementada de forma modular, integra ingesta reproducible (S3 → Parquet → PostgreSQL), indexación vectorial y fusión de rankings, lo que emula patrones de sistemas empresariales a un costo accesible y con operación en CPU para tiempo real (la GPU se limita al backfill de embeddings en procesos offline) .

La demostración sobre una muestra operativa de 15.000 ítems del conjunto FooDI-ML valida la viabilidad técnica: se probó búsqueda híbrida, recuperación multimodal y RAG-to-SQL, con un plan de evaluación holística que va más allá de métricas IR tradicionales (incluyendo Score@K, PWR, POC e indicadores de ganancia híbrida).

En cuanto a la estrategia de **recuperación**, los resultados convergen en:

- **e5** como **motor de candidatos** en escenarios con filtros (maximiza *Recall@K*);
- **BM25** en paralelo y **fusión temprana** para capturar coincidencias literales.
- **GTE** como **señal de coherencia por tienda** en *reranking* (o **retrieve primario** cuando la intención de tienda es explícita).
- **señales de separación** (gte/je3) y **boosts estructurales**;
- **CLIP I→T** como **desempate** sobre el *top-M* cuando hay imagen.

Todo ello se alinea con el **objetivo central** de ofrecer a PYMEs una **arquitectura de referencia open-source** que unifique **búsqueda híbrida**, **multimodalidad** y **RAG-to-SQL** con **viabilidad económica.**

Conclusiones personales

Desde la perspectiva de la autora, la modularidad y la orientación a prototipos rápidos han sido determinantes para acercar capacidades de “gran empresa” al contexto de las PYMEs. Partiendo de un stack 100% abierto (PostgreSQL/pgvector, PyTorch/Transformers, LangChain), fue posible iterar con rapidez, medir con métricas operativas y ajustar el balance calidad–latencia–costo. Un aprendizaje clave es que la GPU no es condición de despliegue: resulta útil en la indexación offline, pero la inferencia del motor puede operar en CPU, reduciendo barreras de entrada para negocios pequeños y medianos. En síntesis, lo híbrido vence a lo monolítico: combinar señales simples y abiertas (BM25 + vectores + estructura) supera a perseguir un único modelo “perfecto” en aislamiento.

Capítulo 7. Futura líneas de trabajo

7.1 Industrialización *open-source* para PYMEs (coste y operación).

El siguiente paso natural es llevar el prototipo a operación con recursos contenidos. Se propone la contenerización y la IaC con Docker Compose como base y una orquestación mínima por particiones (país/tienda) para permitir despliegues repetibles en CPU y autoscaling simple. En paralelo, se implementará observabilidad ligera (logging estructurado, métricas de latencia y Recall@K por faceta) y tableros de hibridación que muestren la proporción BM25 vs. vector, insumos que alimentarán el tuning descrito en 7.3.

7.2 Curación y evaluación con "verdad-terreno".

Sobre esa base operativa, se requiere un marco de evaluación confiable. Se construirá una muestra anotada estratificada por país/tienda para estimar Recall@K y win-rate, y se integrarán pruebas de regresión de IR (Score@K, PWR, POC) al ciclo de CI. Este ground truth permitirá contrastar versiones y cuantificar mejoras de manera estadísticamente sólida, habilitando el tuning controlado de 7.3 y 7.4.

7.3 *Tuning* de recuperación y fusión.

Con el circuito de evaluación consolidado, se optimizarán HNSW por partición (m, efconstruction, efsearch) y se calibrarán escalas de similitud (z-score por dominio). Además, se explorará el aprendizaje de pesos de fusión (RRF ⇄ mezcla) vía un reranker ligero (regresión) entrenado en features abiertas (similitudes e5/gte/je3, señales léxicas y estructurales). Las mejoras se validarán con nDCG@K y Recall@K frente a la línea base definida en 7.2.

7.4 Rerankers *open-source* y señales multimodales.

Para aumentar la precisión en top-N sin comprometer latencia, se incorporarán cross-encoders ligeros como rerankers cuando el presupuesto lo permita. Complementariamente, se reforzará el tie-breaker CLIP I→T con minado de hard negatives, incrementando la separación de pares y el Recall@K multimodal. Los efectos de estas adiciones se medirán con MRR y métricas cruzadas, realimentando los pesos de fusión de 7.3.

7.5 RAG-to-SQL con *guardrails* productivos.

Una vez robustecida la recuperación, se fortalecerá la capa RAG-to-SQL con guardrails de producción: plantillas y schema linking por vertical, restricciones de sintaxis y estimación de costos previa a la ejecución, más validadores del result set. Ante fallos SQL, se activará retroalimentación al retriever (reforzar filtros/facetas) para cerrar el ciclo. Esta línea atiende el reto de interacción semántica-estructurada identificado a lo largo de la tesis.

7.6 Portabilidad y referencia para PYMEs.

Para facilitar adopción en contextos heterogéneos, se publicarán scripts y manifiestos agnósticos a proveedor, preservando PostgreSQL/pgvector como núcleo. La arquitectura de referencia open-source se empaquetará con cookbooks por tamaño de catálogo y guías de quick-start, de modo que equipos pequeños puedan replicar el sistema con mínima fricción.

7.7 Ampliaciones de dominio y datos.

En paralelo a la industrialización, se ampliará la cobertura semántica con normalización de taxonomías y diccionarios de sinónimos por país/tienda. Se consolidará la ingesta incremental con control de calidad (S3 → Parquet → Postgres) y validaciones en el pipeline reproducible existente, lo que debería reflejarse en mejoras de Label Consistency@K e Hybrid Recall en contextos reales.

7.8 Patrón de sistema “empresa-like”, versión PYME.

Finalmente, se consolidará el patrón moderno de búsqueda —candidatos híbridos → fusión → reranking → multimodal → RAG— como paquete reutilizable y documentado. Este empaquetado, nutrido por las prácticas de 7.1–7.7, busca ofrecer a las PYMEs un camino claro y económicamente viable desde el prototipo hasta la operación sostenida.

Referencias

Baymard Institute. (2023). *E-commerce search usability research*. Baymard Institute. Recuperado el 31 de agosto de 2025, de https://baymard.com/research/ecommerce-search

Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. En *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (pp. 757–766). Association for Computing Machinery. https://doi.org/10.1145/1645953.1646084

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75–105. https://doi.org/10.2307/25148625

Jina AI. (2024, 15 de abril). Jina Colbert v1: A new state-of-the-art for search. *Jina AI Blog*. https://jina.ai/news/jina-colbert-v1-a-new-state-of-the-art-for-search/

Khattab, O., & Zaharia, M. (2020). ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. *arXiv*. https://arxiv.org/abs/2004.12832

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. En H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 9459–9474). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

Magnani, B., Seo, J., Hsiao, Y.-C., Verma, S., Satish, S., & Ram, A. (2024). Walmart's semantic search: A case study in large-scale e-commerce search. En *Companion Proceedings of the ACM Web Conference 2024* (pp. 1421–1429). Association for Computing Machinery. https://doi.org/10.1145/3589334.3645558

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv*. https://arxiv.org/abs/1603.09320

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Mohan, R., Guha, R. V., Gupta, V., Sivakumar, D., Hristidis, V., & Kodavalla, H. (2019). Product search at Amazon. En *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2977–2985). Association for Computing Machinery. https://doi.org/10.1145/3292500.3330658

Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2022). MTEB: Massive text embedding benchmark. *arXiv*. https://arxiv.org/abs/2210.07316

Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. *arXiv*. https://arxiv.org/abs/1901.04085

OECD. (2021). *OECD SME and Entrepreneurship Outlook 2021*. OECD Publishing. https://doi.org/10.1787/97a5bbfe-en

Olóndriz, J., Soler, L., Gausach, J., & Treserras, J. (2021). FooDI-ML: A large-scale food dataset for multi-lingual and multi-modal machine learning. En *Proceedings of the 2021 International Conference on Multimedia Retrieval* (pp. 574–578). Association for Computing Machinery. https://doi.org/10.1145/3460426.3463635

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

pgvector team. (2024). *pgvector: Open-source vector similarity search for Postgres* (Version 0.7.2) [Computer software]. GitHub. https://github.com/pgvector/pgvector

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *arXiv*. https://arxiv.org/abs/2103.00020

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *arXiv*. https://arxiv.org/abs/1908.10084

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval, 3*(4), 333–389. https://doi.org/10.1561/1500000019

Salesforce. (2024). *AI, data, CRM media resource*. Salesforce News. https://www.salesforce.com/news/ai-data-crm-media-resource/

Thakur, N., Reimers, N., Rücklé, J., Srivastava, A., & Gurevych, I. (2021). BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. *arXiv*. https://arxiv.org/abs/2104.08663

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. En I. Guyon, U. V. Luxburg, S.

Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), Advances in Neural Information Processing Systems 30 (pp. 5998–6008). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf