Sistema de Acceso a Información Médica Bilingüe Euskera-Castellano mediante RAG

Autora: [Tu Nombre Aquí]

Tutora: [Nombre de la tutora]

Máster en Inteligencia Artificial – Universidad [Nombre de la Universidad]

Junio 2025

Junio 2025

Índice de Contenido

[1. Introducción 5](#_Toc201598451)

[1.1. Justificación y planteamiento del problema 5](#_Toc201598452)

[1.2. Objetivos 6](#_Toc201598453)

[1.2.1. Objetivo general 6](#_Toc201598454)

[1.2.2. Objetivos específicos 6](#_Toc201598455)

[1.3. Metodología 7](#_Toc201598456)

[2. Modelos de embeddings e Indexación vectorial 9](#_Toc201598457)

[2.1. Creación del dataset de evaluación 9](#_Toc201598458)

[2.1.1. Automatización de la generación del Corpus médico 10](#_Toc201598459)

[2.1.2. Generación del Conjunto de Evaluación 14](#_Toc201598460)

[2.2. Creación del dataset para el fine tuning 17](#_Toc201598461)

[2.3. Evaluación y comparativa de modelos de embeddings 17](#_Toc201598462)

[2.4. Fine-tuning del Modelo BGE-M3 para dominio médico 22](#_Toc201598463)

[2.4.1. Motivación y justificación teórica 22](#_Toc201598464)

[2.4.2. Arquitectura y características del modelo base 23](#_Toc201598465)

[2.4.3. Estrategia de fine-tuning 23](#_Toc201598466)

[2.4.4. Dataset de entrenamiento 24](#_Toc201598467)

[2.4.5. Proceso de entrenamiento y resultados 25](#_Toc201598468)

[2.4.6. Evaluación de similitud semántica en términos técnicos vs. Coloquiales 29](#_Toc201598469)

[2.4.6.1. Diseño experimental 29](#_Toc201598470)

[2.4.6.2. Resultados experimentales 30](#_Toc201598471)

[2.5. Pipeline de indexación y almacenamiento vectorial en ChromaDB 33](#_Toc201598472)

[2.5.1. Selección de ChromaDB como sistema de almacenamiento vectorial 33](#_Toc201598473)

[2.5.2. Arquitectura del Pipeline de indexación 34](#_Toc201598474)

[2.5.2.1. Configuración del sistema 34](#_Toc201598475)

[2.5.2.2. Procesamiento multi-fuente 35](#_Toc201598476)

[2.5.2.3. Segmentación (Chunking) 35](#_Toc201598477)

[2.5.2.4. Enriquecimiento semántico con KeyBERT 36](#_Toc201598478)

[2.5.2.4.1. Función personalizada de embeddings 36](#_Toc201598479)

[2.5.2.5. Implementación de colecciones vectoriales diferenciadas 37](#_Toc201598480)

[2.5.2.6. Estructura de almacenamiento en ChromaDB 37](#_Toc201598481)

[2.5.2.7. Validación y monitorización del corpus indexado 39](#_Toc201598482)

[2.5.2.8. Validación Semántica con UMAP 40](#_Toc201598483)

[3. Metodología experimental 45](#_Toc201598484)

[3.1. Introducción 45](#_Toc201598485)

[3.2. Marco metodológico general 46](#_Toc201598486)

[3.3. Arquitectura experimental 47](#_Toc201598487)

[3.3.1. Infraestructura 47](#_Toc201598488)

[3.3.2. Estructura de los resultados 47](#_Toc201598489)

[3.4. Métricas de evaluación 48](#_Toc201598490)

[3.4.1. Métricas principales 48](#_Toc201598491)

[3.4.2. Métricas auxiliares 49](#_Toc201598492)

[3.5. Dataset de evaluación 49](#_Toc201598493)

[3.6. Diseño experimental por métodos 50](#_Toc201598494)

[3.6.1. Condiciones controladas 50](#_Toc201598495)

[3.6.2. Implementación modular 50](#_Toc201598496)

Índice de tablas

[1 Tabla 1.1 – Distribución de consultas por subcategoría en el dataset\_test.json 7](#_Toc201613977)

[2 Tabla 2.1 Modelos evaluados sin fine-tuning 18](#_Toc201613978)

[3 Tabla 2.2 Modelos evaluados. Resultados comparativos 19](#_Toc201613979)

[4 Tabla 2.3 Modelos evaluados con el modelo de finetuning. Resultados comparativos 28](#_Toc201613980)

[5 Tabla 2.4 Comparación Semántica: Modelo Generalista BGE-M3 vs. Versión fine-tuneada en Pares Técnico–Coloquial 31](#_Toc201613981)

[6 Tabla 2.5 Comparación semántica Modelo Generalista BGE-M3 vs. Versión fine-tuneada en Pares de control 33](#_Toc201613982)

[7 Tabla 2.6 Estadísticas del proceso de chunking 39](#_Toc201613983)

[8 Tabla 2.7 Comparativa: Modelo generalista vs. Modelo fine-tuneado 44](#_Toc201613984)

[9 Tabla 3.1 Resultados de evaluación para TF-IDF, Bi-Encoder y Cross-Encoder (modelo generalista BAAI/bge-m3) 62](#_Toc201613985)

[10 Tabla 3.2 Resultados de evaluación para TF-IDF, Bi-Encoder y Cross-Encoder (modelo fine-tuneado bge\_m3\_epochs/epoch4\_MRR0.9717) 62](#_Toc201613986)

Índice de figuras

[1- 2.1 Proceso de creación del Corpus médico. Elaboración propia con apoyo de la herramienta de IA Napkin. 12](#_Toc201527022)

[2- 2.2 Contenido Corpus médico. Elaboración propia con apoyo de la herramienta de IA Napkin. 13](#_Toc201527023)

[3. 2.3 Proyección UMAP de embeddings generados por BAAI/bge-m3 41](#_Toc201527024)

[4- 2.4 Proyección UMAP de embeddings del modelo fine-tuneado (epoch4\_MRR0.9717) 43](#_Toc201527025)

# Introducción

## Justificación y planteamiento del problema

En el contexto actual de salud digital, las personas usuarias buscan información médica fiable en Internet. Aunque lo óptimo sigue siendo acudir a una consulta médica profesional, existe una demanda creciente de recursos informativos accesibles y comprensibles.

Sin embargo, la forma en que se expresan (lenguaje coloquial) difiere considerablemente del lenguaje técnico utilizado por profesionales. Esta brecha semántica representa un reto para los sistemas automáticos de recuperación de información.

A esto se suma el reto del multilingüismo. En comunidades bilingües como la vasca, donde conviven el castellano y el euskera, los sistemas deben ser capaces de entender y responder adecuadamente en ambos idiomas.

Las lenguas minorizadas como el euskera presentan barreras tecnológicas importantes. Durante el desarrollo de este TFM no se encontraron modelos de generación ni embeddings que ofrecieran resultados satisfactorios, lo que obligó a tomar decisiones técnicas alternativas, como el uso de traducción automática para procesar consultas y generar respuestas.

Esta decisión añadió una capa de complejidad técnica y un aumento en el coste computacional, pero permitió mantener el enfoque multilingüe del sistema.

Durante los experimentos se probaron motores de búsqueda tradicionales basados en palabras clave, como TF-IDF y BM25 combinados con Bi-Encoder. Si bien estos enfoques son eficientes y ampliamente usados, mostraron limitaciones al trabajar con expresiones informales o variantes lingüísticas presentes en el corpus médico. Por esta razón, se exploró el uso de arquitecturas más sofisticadas como Retrieval-Augmented Generation (RAG), que permiten combinar búsquedas semánticas con generación de respuestas personalizadas y contextualizadas.

La inspiración inicial para este trabajo surgió al visitar el portal [Osasun Eskola](https://www.osakidetza.euskadi.eus/tu-portal-de-salud-y-vida-sana/webosk00-oskenf/es/), un servicio de Osakidetza y del Departamento de Salud del Gobierno Vasco que proporciona formación e información médica a la ciudadanía, con el fin de fomentar una actitud activa y responsable hacia la salud y la enfermedad.

Sin embargo, navegar por esta web puede ser complejo para personas no expertas, lo que motivó el desarrollo de un asistente conversacional accesible.

El objetivo fue construir un asistente médico informativo que permitiera acceder fácilmente a información médica fiable en un formato accesible, interactivo y en ambos idiomas oficiales. Un sistema que no sustituye la consulta médica, pero que apoya a la ciudadanía con recomendaciones útiles y comprensibles, siguiendo el espíritu de la Osasun Eskola.

## Objetivos

## Objetivo general

Desarrollar un sistema de Recuperación Aumentada con Generación (RAG) que permita responder consultas médicas informativas en castellano y euskera, combinando técnicas de búsqueda semántica, traducción automática, generación de lenguaje y memoria conversacional.

## Objetivos específicos

* Evaluar distintos modelos de embeddings multilingües para la creación de una base de datos vectorial.
* Realizar fine-tuning sobre un modelo generalista (BAAI/bge-m3) usando pares de preguntas/respuestas médicas tanto técnicas como en lenguaje coloquial.
* Comparar estrategias de recuperación: TF-IDF, BM25, Bi-Encoder, Cross-Encoder, pipelines híbridos.
* Integrar un sistema conversacional multilingüe con memoria y traducción automática. •
* Desplegar una interfaz sencilla mediante Streamlit.

## Metodología

Se ha adoptado una metodología modular basada en fases:

* **Preparación del corpus**: extracción, limpieza y segmentación de documentos médicos multimodales (PDFs de diversas fuentes sanitarias y contenido web de Osakidetza).
* **Construcción de datasets de evaluación:**
  + **dataset\_test.json:** 322 queries médicas con ground truth manual, abarcando 8 categorías clínicas (neurología, cáncer, cardiología, endocrinología, respiratorias, salud mental, enfermedades comunes, ITS) y 18 subcategorías especializadas (diabetes, ictus, cáncer de mama, VIH/SIDA, EPOC, esquizofrenia, etc.)

Todo el código fuente desarrollado durante este Trabajo Fin de Máster se encuentra disponible públicamente en el repositorio de GitHub: <https://github.com/Nagore2021/TFM>, con el fin de fomentar la reproducibilidad, la transparencia y el acceso abierto al conocimiento generado.

1 Tabla 1.1 – Distribución de consultas por subcategoría en el dataset\_test.json

| **Categoría** | **Subcategoría** | **Nº de Preguntas** |
| --- | --- | --- |
| Neurología | Ictus | 12 |
| Neurología | Epilepsia | 15 |
| Enfermedades comunes | Gripe | 14 |
| Cáncer | Cáncer colorrectal | 26 |
| Cáncer | Cáncer de mama | 27 |
| Cáncer | Cáncer de pulmón | 7 |
| Cardiología | Angina / Infarto | 5 |
| Cardiología | Hipertensión arterial | 13 |
| Cardiología | Mareo / Síncope / Parada | 2 |
| ITS | VIH / SIDA | 28 |
| Endocrinología | Diabetes | 25 |
| Respiratorias | Bronquiolitis | 18 |
| Respiratorias | EPOC | 21 |
| Salud mental | Agorafobia | 14 |
| Salud mental | Esquizofrenia | 26 |
| Salud mental | Depresión | 27 |
| Salud mental | Trastorno bipolar | 23 |
| Salud mental | Ansiedad | 19 |

*Fuente: Elaboración propia a partir del dataset\_test.json. Total: 322 consultas anotadas manualmente.*

* + **dataset\_finetune.json:** >2000 pares pregunta-respuesta médicas (lenguaje coloquial ↔ terminología técnica)
* **Evaluación y selección de modelos de embeddings:** comparativa sistemática de modelos generalistas (BAAI/bge-m3, Qwen3-Embedding-8B) y especializados (bsc-bio-ehr-es, RigoBERTa-Clinical) mediante métricas de recuperación (Precision@K, Recall@K, F1@K, MRR).
* **Fine-tuning médico especializado:** adaptación del modelo BAAI/bge-m3 con datos médicos del dominio para mejorar la captura de relaciones semánticas específicas.
* **Pre-Retrieval**: indexación vectorial en ChromaDB con dos colecciones independientes (modelo base BAAI/bge-m3 vs. modelo fine-tuneado) y enriquecimiento de metadatos médicos.
* **Retrieval:** implementación y evaluación comparativa de estrategias híbridas (TF-IDF, BM25, Bi-Encoder, Cross-Encoder).
* **Generación:** integración con modelos de lenguaje, traducción automática bidireccional euskera-castellano y memoria conversacional.
* **Producción:** despliegue del sistema RAG completo con interfaz conversacional Streamlit.

# Modelos de embeddings e Indexación vectorial

## Creación del dataset de evaluación

Con el objetivo de desarrollar un sistema RAG que proporcione respuestas informativas en el ámbito de la salud, se recopiló un corpus médico multilingüe relevante, proveniente principalmente de fuentes oficiales como el portal Osasun Eskola de Osakidetza y otros organismos sanitarios autonómicos. Se seleccionaron 8 categorías clínicas clave: neurología, cáncer, cardiología, endocrinología, respiratorias, salud mental, enfermedades comunes e ITS. Cada categoría incluye subcategorías específicas (como ictus, VIH/SIDA, diabetes o cáncer de mama), tal y como se muestra en la Tabla 1.1 del Capítulo 1

Dado que el portal Osasun Eskola no ofrecía suficientes documentos PDF descargables, se completó el corpus con información procedente de otras fuentes institucionales fiables. Algunos ejemplos incluyen:

* Agorafobia: Gobierno Vasco, Departamento de Salud (Guías de Educación Sanitaria).
* Trastorno Bipolar: Gobierno de Aragón, Departamento de Salud y Familia.
* Cáncer de mama: Federación Española de Cáncer de Mama (FEDER), Pierre Fabre.
* Depresión: Confederación Salud Mental España, SEMERGEN, Guía Salud (Gobierno de Aragón).
* Ictus: Sociedad Valenciana de Neurología.

Para estructurar el corpus, se definieron dos archivos JSON de metadatos:

* **pdf\_metadata.json**: contiene información sobre los documentos en PDF, incluyendo nombre del archivo, categoría, subcategoría, idioma, fuente y palabras clave.
* **web\_metadata.json**: recopila URLs del portal Osasun Eskola con su correspondiente categorización, idioma y términos clave.

Ambos JSON comparten la siguiente estructura de metadatos:

{

*"****filename****": "consejos\_gripe\_general\_es.pdf",*

*"****categoria****": "enfermedades\_comunes",*

*"****subcategoria****": "Gripe",*

*"****idioma****": "es",*

*"****publico\_objetivo****": "general",*

*"****fuente****": "PDF",*

*"palabras\_clave": ["gripe", "fiebre alta", "virus", "escalofríos"]*

*}*

## Automatización de la generación del Corpus médico

Para facilitar la reutilización del sistema y permitir futuras ampliaciones, se desarrolló un script automatizado (**1\_crear\_corpus.py**) encargado de construir el corpus completo a partir de los archivos de metadatos **pdf\_metadata.json** y **web\_metadata.json**. Este enfoque garantiza escalabilidad, trazabilidad y consistencia en el procesamiento de los documentos.

**Extracción de contenido web:**

Con el fin de extraer información desde el portal Osasun Eskola, se realizó un análisis previo de su estructura HTML para identificar los patrones comunes de organización del contenido. Posteriormente, se empleó la librería **BeautifulSoup** para implementar técnicas de web scraping, limpiando el contenido de etiquetas innecesarias, menús o elementos decorativos. Se prestó especial atención a la extracción semántica del cuerpo informativo principal de cada página.

**Procesamiento de documentos PDF:**

Para los documentos descargados en formato PDF, se utilizó la librería PyMuPDF (import fitz) por su robustez y eficiencia en la extracción de texto desde archivos con estructuras variadas. Se implementó un proceso de normalización textual que incluyó:

* Eliminación de caracteres Unicode no imprimibles.
* Supresión de saltos de línea intermedios innecesarios.
* Corrección de guiones de separación al final de línea.
* Conversión homogénea de espacios múltiples.

Se optó por **PyMuPDF** frente a otras librerías (como PDFplumber o pdfminer) debido a su superior rendimiento en PDFs con estructuras complejas y su capacidad para preservar el formato textual original. Para el web scraping, **BeautifulSoup** proporcionó la flexibilidad necesaria para adaptarse a las diferentes estructuras del portal Osasun Eskola.

**Flujo general del script:**

El script realiza las siguientes operaciones:

* **Carga de metadatos desde los archivos JSON** previamente definidos, con información estructurada sobre categoría clínica, idioma, tipo de fuente, público objetivo y palabras clave.
* **Extracción automática de contenido** mediante funciones específicas:
  + extract\_pdf\_text() para leer y limpiar texto desde documentos PDF.
  + extract\_web\_text() para extraer y normalizar contenido HTML desde URLs del portal Osakidetza.
* **Almacenamiento final** en un archivo unificado **corpus\_medico.json**, que contiene:
  + **corpus**: diccionario con los textos completos.
  + **metadata**: metadatos asociados a cada documento.
  + **stats**: resumen estadístico del corpus.

Gráfico, Gráfico de embudo

El contenido generado por IA puede ser incorrecto.

1- 2.1 Proceso de creación del Corpus médico. Elaboración propia con apoyo de la herramienta de IA Napkin.

**Estadísticas y ejemplo del corpus generado**

El resultado del proceso fue un corpus médico con las siguientes estadísticas:

* **Total de documentos:** 45
* **Documentos PDF:** 17
* **Documentos Web:** 28
* **Total de caracteres procesados:** 1.742.774

Imagen de la pantalla de un celular con letras

El contenido generado por IA puede ser incorrecto.

2- 2.2 Contenido Corpus médico. Elaboración propia con apoyo de la herramienta de IA Napkin.

A continuación, se muestra un fragmento representativo del archivo corpus\_medico.json:

*{*

*"****corpus****": {*

*"pdf\_diabetes\_cuidarse\_es": "Actividad física• Realizar una actividad física adecuada a tu capacidad...",*

*"web\_cancer-de-mama": "¿Qué es el cáncer de mama?Es una enfermedad que afecta casi exclusivamente a las mujeres..."*

*},*

*"****metadata****": {*

*"****pdf\_diabetes\_cuidarse\_es****": {*

*"filename": "diabetes\_cuidarse\_es.pdf",*

*"categoria": "diabetes",*

*"subcategoria": "diabetes",*

*"idioma": "es",*

*"publico\_objetivo": "general",*

*"fuente": "PDF",*

*"palabras\_clave": ["diabetes", "cuidados"]*

*},*

*"****web\_cancer-de-mama****": {*

*"url": "https://www.osakidetza.euskadi.eus/cancer-de-mama/webosk00-oskenf/es/",*

*"categoria": "cancer",*

*"subcategoria": "cancer de mama",*

*"idioma": "es",*

*"publico\_objetivo": "general",*

*"fuente": "Web",*

*"palabras\_clave": ["cancer mama", "prevencion", "diagnostico", "tratamiento", "mamografia"]*

*}*

*},*

*"****stats****": {*

*"total\_documents": 45,*

*"pdf\_documents": 17,*

*"web\_documents": 28,*

*"total\_characters": 1742774*

*}*

*}*

## Generación del Conjunto de Evaluación

Adicionalmente, se elaboró manualmente un archivo **test\_queries.json** que contiene 110 consultas de prueba diseñadas para evaluar el sistema RAG.

Estas preguntas están formuladas tanto en lenguaje médico como en lenguaje coloquial y abarcan distintas categorías y subcategorías clínicas. Cada consulta incluye información sobre la dificultad, la categoría, la subcategoría y los documentos considerados relevantes. Esta base resulta fundamental para realizar una evaluación controlada de los distintos métodos de recuperación implementados posteriormente, asegurando una cobertura temática y lingüística representativa del dominio clínico.

A continuación, se muestra un ejemplo representativo del contenido del archivo **test\_queries.json**:

*{*

*"queries": [ {*

*"text\_es": "¿Depresión en la adolescencia?",*

*"text\_eu": "",*

*"relevant\_docs": [*

*"pdf\_depresion\_juvenil\_es",*

*"pdf\_depresion\_adolescencia\_es",*

*"web\_depresion",*

*"pdf\_depresion\_guia\_es",*

*"pdf\_depresion\_pacientes\_es"*

*],*

*"categoria": "salud mental",*

*"subcategoria": "depresion",*

*"dificultad": "dificil"*

*},*

*{*

*"text\_es": "Mi padre oye voces que no existen",*

*"text\_eu": "Nire aitak ez dauden ahotsak entzuten ditu",*

*"relevant\_docs": ["web\_esquizofrenia"],*

*"categoria": "psiquiatria",*

*"subcategoria": "esquizofrenia",*

*"dificultad": "alta"*

*}]}*

Finalmente, para fusionar de forma automática el contenido del corpus y las consultas de prueba en un único archivo que sirva como dataset de evaluación, se creó el script **2\_generar\_dataset\_test.py**. Este script genera el archivo dataset\_test.json combinando la información del corpus (corpus\_medico.json) con las queries del archivo test\_queries.json. De este modo, se obtiene un único JSON estructurado que será utilizado de forma consistente en todas las fases del proyecto para evaluar el desempeño de los distintos modelos de recuperación.

**Ejemplo del archivo dataset\_test.json**

{

"queries": [

{

"text\_es": "Mi padre oye voces que no existen",

"text\_eu": "Nire aitak ez dauden ahotsak entzuten ditu",

"relevant\_docs": ["web\_esquizofrenia"],

"categoria": "salud mental",

"subcategoria": "esquizofrenia",

"dificultad": "alta"

}

],

"corpus": {

"web\_esquizofrenia": "¿Qué es?La Esquizofrenia es una enfermedad mental que afecta al pensamiento, la conducta (forma de comportarse), las emociones y la manera de percibir las cosas. Pertenece al grupo de los trastornos psicóticos, lo que significa que las personas con esquizofrenia pueden tener dificultades para distinguir entre la realidad y los pensamientos, las ideas o las percepciones producidos por la enfermedad..."

}}

Este dataset\_test.json se ha ido ampliando hasta tener 322 consultas, tal y como se muestra en la Tabla 1.1 del Capítulo 1

## Creación del dataset para el fine tuning

Se elaboró manualmente un dataset (dataset\_finetune.json) con 2.033 pares pregunta-respuesta en castellano para el fine-tuning de modelo de embedding Las consultas fueron diseñadas para cubrir las 8 categorías clínicas principales del corpus, combinando tanto terminología médica especializada como lenguaje coloquial típico de pacientes.

*[ {"query": "¿Por qué los diabéticos deben evitar el tabaco?",*

*"response": "El tabaco potencia el daño vascular, aumentando el riesgo de infarto, ictus y complicaciones en pies y riñones”},*

*{"query": "¿Por qué los niños pequeños son más propensos a bronquiolitis?",*

*"response": "Sus vías respiratorias son más estrechas y su sistema inmunitario aún inmaduro, lo que facilita la obstrucción y la infección."*

*}]*

## Evaluación y comparativa de modelos de embeddings

**Objetivo de la evaluación de modelos sin fine-tuning**

Esta evaluación inicial tiene como **objetivo fundamental identificar el modelo base más prometedor** para servir como foundation en las siguientes fases del desarrollo. No se busca el modelo final del sistema, sino **determinar qué arquitectura de embedding ofrece el mejor punto de partida** para posteriores optimizaciones como fine-tuning y estrategias híbridas.

La selección se centró en modelos **específicamente diseñados para tareas de embedding y retrieval**, con diferentes niveles de complejidad y especialización, para identificar la arquitectura más prometedora como base del sistema RAG médico.

En esta fase únicamente se evaluaron modelos sin fine-tuning para identificar la arquitectura base más prometedora.

A continuación se presentan los modelos seleccionados:

2 Tabla 2.1 Modelos evaluados sin fine-tuning

| **Modelo** | **Tipo** | **Especialización** | **Dimensión** |
| --- | --- | --- | --- |
| BAAI/bge-m3 | Generalista | Multilingüe, retrieval | 1024 |
| Qwen/Qwen3-Embedding-8B | Generalista | Última generación | 8192 |
| sentence-transformers/all-MiniLM-L6-v2 | Generalista | Compacto | 384 |
| jinaai/jina-embeddings-v2-base-es | Generalista | Multilingüe | 768 |

**Metodología de evaluación**

Los modelos fueron evaluados mediante un script automatizado (**3\_evaluar\_modelos.py**) que implementa un pipeline completo de evaluación:

**Dataset de evaluación**: Se utilizó dataset\_test.json, que contiene consultas médicas reales con sus correspondientes documentos de referencia (ground truth). Las consultas incluyen texto tanto en español como en euskera, aplicándose traducción automática con el modelo Helsinki-NLP/opus-mt-eu-es cuando no existe la consulta en español.

**Implementación técnica**

La evaluación se realizó mediante un script automatizado que utiliza una función simplificada para generar embeddings:

*def get\_embeddings(model, texts, batch\_size=16):*

*if not texts:*

*return torch.tensor([])*

*return model.encode(texts, batch\_size=batch\_size, convert\_to\_tensor=True)*

Esta implementación directa aprovecha que todos los modelos evaluados pertenecen a la familia SentenceTransformers y poseen el método .encode() optimizado, garantizando que cada modelo utilice su estrategia de pooling específica (CLS pooling para BGE-M3, pooling variable para Qwen3, etc.).

**Proceso de evaluación:**

Para cada modelo se generan embeddings del corpus completo y de las consultas, se calcula similitud coseno entre consultas y documentos, y se extraen los top-K resultados más similares para el cálculo de métricas.

**Métricas empleadas:**

* **Precision@K**: Proporción de documentos relevantes entre los K primeros resultados recuperados
* **Recall@K**: Proporción de documentos relevantes recuperados respecto al total de documentos relevantes
* **F1@K**: Media armónica entre precisión y recall
* **MRR (Mean Reciprocal Rank)**: Recíproco de la posición del primer resultado relevante

**Resultados comparativos**

Los resultados obtenidos para K=3 se muestran en la siguiente tabla:

3 Tabla 2.2 Modelos evaluados. Resultados comparativos

| **Modelo** | **Precision@3** | **Recall@3** | **F1@3** | **MRR** |
| --- | --- | --- | --- | --- |
| Qwen/Qwen3-Embedding-8B | **0.3364** | 0.7237 | **0.4387** | **0.7116** |
| BAAI/bge-m3 | 0.3354 | **0.7256** | 0.4375 | 0.7003 |
| sentence-transformers/all-MiniLM-L6-v2 | 0.2723 | 0.5901 | 0.3541 | 0.5870 |
| jinaai/jina-embeddings-v2-base-es | **0.0311** | **0.0702** | **0.0404** | **0.1233** |

**Análisis de resultados por tipo de modelo**

* **Modelos generalistas top-tier (Qwen, BGE-M3)**
* Rendimiento: Lideran todas las métricas con diferencias mínimas entre ellos
* Fortaleza: Entrenamiento específico para retrieval con datasets masivos y diversos
* Estrategias de pooling especializadas por modelo:
  + **BGE-M3**: Implementa CLS pooling donde el token especial [CLS] es entrenado específicamente durante el preentrenamiento para actuar como agregador semántico de la secuencia completa.
  + **Qwen3-Embedding-8B**: Utiliza pooling optimizado con dimensionalidad variable mediante el framework SentenceTransformers. Incorpora una capa de transformación lineal configurable que permite proyectar los embeddings internos (4096D) a cualquier dimensión objetivo (32-4096D): output = Linear(internal\_embedding, target\_dim). Esta arquitectura permite ajustar el compromiso entre precisión semántica y eficiencia computacional según los requisitos específicos de la tarea
* **Caso especial: Jina Embeddings**
  + **Rendimiento:** Peor de todos los modelos evaluados (0.0311 en Precision@3)
  + **Paradoja**: Es un modelo específicamente diseñado para embeddings, pero falla completamente
  + **Posibles causas**: Incompatibilidad con español médico o datos de entrenamiento inadecuados para el dominio
* **Modelo all-MiniLM-L6-v2**
* **Rendimiento**: Intermedio, superando a todos los biomédicos
* **Ventaja**: Buen balance entre eficiencia (384 dimensiones) y rendimiento

**Selección del modelo ganador**

Basándose en los resultados obtenidos, se seleccionó **BAAI/bge-m3** como modelo base para las siguientes fases del desarrollo, superando incluso a Qwen/Qwen3-Embedding-8B por las siguientes razones técnicas y prácticas:

**Criterios de decisión**:

1. **Rendimiento competitivo**: Diferencias mínimas con Qwen en todas las métricas (< 3% en MRR)
2. **Eficiencia computacional**: 568M parámetros vs. 8B de Qwen, permitiendo ejecución eficiente en CPU
   1. **Qwen3-Embedding-8B**: 8.000 millones de parámetros
   2. **BAAI/bge-m3**: 568 millones de parámetros

**Ratio**: Qwen es ~14 veces más grande que BGE-M3

## Fine-tuning del Modelo BGE-M3 para dominio médico

## Motivación y justificación teórica

En el contexto médico, donde el vocabulario técnico, las relaciones semánticas específicas y la terminología especializada son críticos, el fine-tuning puede proporcionar ventajas significativas sobre modelos pre-entrenados.

La hipótesis central que motiva este enfoque es que un modelo generalista robusto como BGE-M3, con fine-tuning dirigido en datos médicos en español, puede superar tanto a su versión base como a modelos biomédicos especializados, especialmente cuando se dispone de datasets de entrenamiento limitados pero bien curados.

Las principales razones que motivan esta decisión son:

* **Transferencia de conocimiento**: BGE-M3 posee un conocimiento base amplio sobre relaciones semánticas y sintácticas que puede ser refinado para el dominio médico.
* **Eficiencia computacional**: La adaptación de un modelo pre-entrenado requiere menos recursos que el entrenamiento desde cero

**Limitaciones del enfoque:**

1. **Dataset limitado**: 2,033 ejemplos de pares de pregunta /respuesta vs. millones en el entrenamiento original de BGE-M3
2. **Cobertura especializada**: Existe una concentración temática en ciertas especialidades médicas (neurología, cardiología, endocrinología), lo que limita la capacidad del modelo para generalizar a otras áreas.
3. **Contexto regional**: Terminol.ogía médica española puede no generalizar globalmente
4. **Dependencia de calidad**: La calidad y representatividad de los ejemplos es crucial, ya que el fine-tuning es sensible a errores, ambigüedades o sesgos presentes en los datos.

## Arquitectura y características del modelo base

BGE-M3 es un modelo desarrollado por Beijing Academy of Artificial Intelligence, basado en transformadores y diseñado específicamente para generar embeddings útiles en recuperación de información.

Sus características técnicas principales son:

* Parámetros: 568 millones, equilibrando capacidad y eficiencia.
* Dimensión de embedding: 1024 dimensiones.
* Secuencias: Capacidad de procesar hasta 8192 tokens.
* Pooling: Usa token [CLS] como representación semántica del texto.

Estas propiedades lo hacen especialmente adecuado para tareas de búsqueda en textos largos, como documentos médicos.

## Estrategia de fine-tuning

**Metodología de Entrenamiento**

El fine-tuning se implementó siguiendo las mejores prácticas establecidas para modelos de embeddings en sistemas RAG (Yilmaz, 2024), utilizando la librería SentenceTransformers con un enfoque de aprendizaje mediante la función de pérdida **MultipleNegativesRankingLoss**.

Esta función es especialmente efectiva para tareas de matching semántico, ya que:

* Optimiza la similitud entre pares (query, documento relevante)
* Minimiza la similitud con documentos no relevantes del batch
* Permite un aprendizaje eficiente con ejemplos negativos implícitos

**Configuración de Hiperparámetros**

La configuración se optimizó específicamente para el modelo BGE-M3, considerando su tamaño (568M parámetros) y características arquitectónicas, siguiendo las recomendaciones para fine-tuning de modelos de embeddings en aplicaciones RAG:

**Max sequence length**: 512 tokens Reducido de 8192 para estabilidad

**Learning rate**: 2e-5 Muy conservador para preservar conocimiento

**Batch size**: 16 Reducido para modelo grande

**Épocas máximas**: 10

**Early stopping patience**: 3 Estricto para evitar sobreajuste

**Warmup steps**: 200

**Función de pérdida**: MultipleNegativesRankingLoss

**Métrica de validación**: Mean Reciprocal Rank (MRR)

**Pooling**: CLS token (optimizado para retrieval)

## Dataset de entrenamiento

El dataset de fine-tuning contiene 2.033 pares pregunta-respuesta, con lenguaje técnico y coloquial en el ámbito médico. Fue dividido en:

* Entrenamiento: 80% (1.626 pares)
* Validación: 20% (407 pares)

**Ejemplo técnico:**

*{*

*"query": "¿Cuáles son los criterios diagnósticos para diabetes mellitus tipo 2 según ADA?",*

*"response": "Los criterios incluyen: HbA1c ≥6.5%, glucemia en ayunas ≥126 mg/dL..."*

*}*

**Ejemplo coloquial:**

*{*

*"query": "¿Tener el azúcar alto significa que soy diabético?",*

*"response": "No necesariamente. Una glucemia elevada puede indicar prediabetes o diabetes..."*

*}*

## Proceso de entrenamiento y resultados

El entrenamiento se ejecutó mediante un script especializado (**finetuning\_bge.py**) que implementa una arquitectura específica para BGE-M3:

***Configuración específica del modelo BGE-M3***

*word\_embedding\_model = models.Transformer( "BAAI/bge-m3", max\_seq\_length=512 # Limitado para estabilidad )*

***BGE-M3 utiliza CLS pooling optimizado para retrieval***

*pooling\_model = models.Pooling( word\_embedding\_model.get\_word\_embedding\_dimension(), pooling\_mode\_cls\_token=True, CLS pooling específico para BGE pooling\_mode\_mean\_tokens=False Desactivar mean pooling )*

*model = SentenceTransformer(modules=[word\_embedding\_model, pooling\_model])*

***Configuración de entrenamiento conservadora***

train\_dataloader = DataLoader(train\_data, shuffle=True, batch\_size=16)

train\_loss = losses.MultipleNegativesRankingLoss(model=model)

*# Entrenamiento con early stopping personalizado*

for epoch in range(1, max\_epochs + 1):

model.fit(

train\_objectives=[(train\_dataloader, train\_loss)],

epochs=1,

warmup\_steps=200

if epoch == 1 else 0,

use\_amp=False, *# Desactivar mixed precision para estabilidad*

optimizer\_params={

"lr": 2e-5,

"weight\_decay": 0.01 } )

**Evaluación Personalizada para BGE-M3**

Se implementó una función de evaluación especializada que considera las características específicas de BGE-M3:

***def evaluate\_model\_improved****(model: SentenceTransformer, val\_data: list) -> dict:*

*"""*

*Evaluación optimizada para BGE-M3 con normalización de embeddings*

*"""*

*k = 3*

*query\_to\_relevant = defaultdict(set)*

*all\_responses = set()*

*# Agrupar datos por pregunta*

*for ex in val\_data:*

*query\_text = ex.texts[0]*

*relevant\_text = ex.texts[1]*

*query\_to\_relevant[query\_text].add(relevant\_text)*

*all\_responses.add(relevant\_text)*

*precision\_scores, recall\_scores, f1\_scores, mrr\_scores = [], [], [], []*

*for query\_text in query\_to\_relevant.keys():*

*gold\_docs = list(query\_to\_relevant[query\_text])*

*# Crear corpus con documentos negativos*

*all\_other\_responses = list(all\_responses - set(gold\_docs))*

*negative\_docs = random.sample(all\_other\_responses, min(30, len(all\_other\_responses)))*

*corpus\_docs = gold\_docs + negative\_docs*

*random.shuffle(corpus\_docs)*

*#* ***BGE-M3 con normalización de embeddings***

*query\_emb = model.encode([query\_text], normalize\_embeddings=True,*

*show\_progress\_bar=False)[0]*

*corpus\_embs = model.encode(corpus\_docs, normalize\_embeddings=True,*

*show\_progress\_bar=False)*

***# Similitud coseno optimizada***

*similarities = np.dot(corpus\_embs, query\_emb)*

*ranked\_indices = np.argsort(similarities)[::-1]*

***# Calcular métricas de ranking***

*top\_k\_docs = [corpus\_docs[i] for i in ranked\_indices[:k]]*

*tp = sum(1 for doc in top\_k\_docs if doc in gold\_docs)*

*precision = tp / len(top\_k\_docs) if len(top\_k\_docs) > 0 else 0*

*recall = tp / len(gold\_docs) if len(gold\_docs) > 0 else 0*

*f1 = 2 \* (precision \* recall) / (precision + recall) if (precision + recall) > 0 else 0*

***# Mean Reciprocal Rank***

*mrr = 0.0*

*for i, idx in enumerate(ranked\_indices):*

*if corpus\_docs[idx] in gold\_docs:*

*mrr = 1.0 / (i + 1)*

*break*

*precision\_scores.append(precision)*

*recall\_scores.append(recall)*

*f1\_scores.append(f1)*

*mrr\_scores.append(mrr)*

*return {*

*"Precision@3": round(np.mean(precision\_scores), 4),*

*"Recall@3": round(np.mean(recall\_scores), 4),*

*"F1@3": round(np.mean(f1\_scores), 4),*

*"MRR": round(np.mean(mrr\_scores), 4),*

*"num\_queries\_evaluated": len(precision\_scores)*

*}*

**Resultados del entrenamiento**

4 Tabla 2.3 Modelos evaluados con el modelo de finetuning. Resultados comparativos

| **Modelo** | **Precision@3** | **Recall@3** | **F1@3** | **MRR** |
| --- | --- | --- | --- | --- |
| Qwen/Qwen3-Embedding-8B | **0.3364** | 0.7237 | **0.4387** | **0.7116** |
| BAAI/bge-m3 | 0.3354 | **0.7256** | 0.4375 | 0.7003 |
| ***Fine-tuned: bge\_m3\_epochs/epoch4\_MRR0.9717*** | 0.3520 | 0.7551 | 0.4571 | 0.6806 |
| sentence-transformers/all-MiniLM-L6-v2 | 0.2723 | 0.5901 | 0.3541 | 0.5870 |
| jinaai/jina-embeddings-v2-base-es | **0.0311** | **0.0702** | **0.0404** | **0.1233** |

**Métricas del Modelo Final**

**Modelo seleccionado:** bge\_m3\_epochs/epoch4\_MRR0.9717

Precision@3: 0.8947

Recall@3: 0.7821

F1@3: 0.8297

MRR: 0.9717

## Evaluación de similitud semántica en términos técnicos vs. Coloquiales

Con el objetivo de comprobar si el proceso de fine-tuning había mejorado realmente la capacidad del modelo para capturar relaciones semánticas relevantes en el ámbito médico, se diseñó un experimento comparativo (comparar\_similitud.py) entre el modelo original BAAI/bge-m3 y su versión ajustada (bge\_m3\_epochs/epoch4\_MRR0.9717).

Aunque las métricas de entrenamiento (como MRR o Recall@K) indicaban una convergencia adecuada durante el fine-tuning, no eran suficientes para responder una cuestión clave: **¿ha aprendido el modelo a vincular mejor la terminología médica técnica con el lenguaje cotidiano utilizado por pacientes?**

Por ejemplo, un paciente podría referirse a la diabetes como “azúcar alto”. La pregunta es: **¿el modelo fine-tuneado ha mejorado su capacidad para reconocer esta equivalencia semántica?**

## Diseño experimental

Se implementó un experimento de comparación A/B donde ambos modelos (base vs. fine-tuneado) procesan los mismos pares de términos y se calcula la **similitud coseno** entre sus embeddings. Esta métrica, ampliamente utilizada en procesamiento de lenguaje natural, nos indica qué tan "cercanos" considera el modelo que están dos conceptos en su espacio de representación vectorial.

Para garantizar la validez de los resultados, se diseñaron dos conjuntos de pares complementarios.

**Conjunto objetivo - Pares médicos relacionados (11 pares)** Estos representan equivalencias semánticas reales entre terminología clínica y lenguaje coloquial:

* + "diabetes" ↔ "azúcar alto" (endocrinología)
  + "depresión" ↔ "tristeza" (psiquiatría)
  + "cáncer" ↔ "tumor" (oncología)
  + "ictus" ↔ "interrupción repentina del flujo sanguíneo en el cerebro" (neurología)

***Hipótesis:*** Si el fine-tuning ha sido efectivo, la similitud entre estos pares debería aumentar.

**Conjunto control - Pares médicos no relacionados (8 pares)**

Este conjunto incluye términos médicos que aparecen en el dataset de fine-tuning, pero sin una relación clínica entre sí..

* + "cáncer" ↔ "asma"
  + epilepsia ↔ alergia

***Hipótesis:*** Estos pares no deberían mostrar mejoras significativas, ya que no están semánticamente relacionados.

## Resultados experimentales

Se analizaron 19 pares en total (11 médicos y 8 de control). A continuación, se muestran los resultados obtenidos:

**Análisis de pares médicos relacionados**

5 Tabla 2.4 Comparación Semántica: Modelo Generalista BGE-M3 vs. Versión fine-tuneada en Pares Técnico–Coloquial

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Par evaluado** | **Modelo base** | **Modelo fine-tuneado** | **Diferencia** | **Variación (%)** |
| diabetes ↔ azúcar alto | 0.646 | 0.636 | -0.010 | -1.5% |
| cáncer colorrectal ↔ tumor maligno | 0.603 | 0.633 | +0.030 | **+5.0%** |
| cáncer ↔ tumor | 0.803 | 0.876 | +0.073 | **+9.1%** |
| cáncer de colon ↔ colonoscopia | 0.693 | 0.657 | -0.036 | -5.2% |
| recidiva del cáncer de mama ↔ reaparición del cáncer | 0.746 | 0.818 | +0.071 | +**9.5%** |
| cáncer de mama ↔ mamografía | 0.650 | 0.652 | +0.002 | **+0.3%** |
| depresión ↔ tristeza | 0.617 | 0.724 | +0.106 | **+17.2%** |
| HbA1c ↔ promedio de glucosa en sangre | 0.437 | 0.393 | -0.044 | -10.0% |
| cáncer de pulmón ↔ tos persistente | 0.423 | 0.167 | -0.256 | -60.5% |
| hipoglucemia ↔ bajada de azúcar | 0.670 | 0.776 | +0.106 | **+15.8%** |
| ictus ↔ interrupción repentina del flujo sanguíneo en el cerebro | 0.503 | 0.437 | -0.066 | -13.2% |

**Casos donde el fine-tuning mejora**

Se detecta una mejora clara en **6 de 11 pares** (54.5%), especialmente en relaciones claras y frecuentes como:

* depresión ↔ tristeza (+17.2%)
* hipoglucemia ↔ bajada de azúcar (+15.8%)
* cáncer ↔ tumor (+9.1%)

Los pares que más mejoran comparten características:

* **Correspondencia directa**: "depresión" y "tristeza" son conceptualmente muy próximos
* **Frecuencia en consultas**: Términos que los pacientes usan habitualmente
* **Claridad semántica**: Relaciones sin ambigüedad conceptual

**Casos sin mejora o con deterioro**

No todos los pares obtuvieron mejoras:

* + **"cáncer pulmón" ↔ "tos persistente"**: Deterioro del 60.5% (0.423 → 0.157). El modelo puede haber aprendido correctamente que estos términos están *relacionados* pero no son *equivalentes*. La tos persistente es un síntoma, no un sinónimo de cáncer de pulmón.
  + **HbA1c ↔ Promedio de glucosa en sangre y ictus ↔** interrupción del flujo cerebral también presentan descensos, lo que podría deberse a que estas equivalencias no aparecen con suficiente frecuencia o claridad en los datos usados para el fine-tuning.Este es un término muy técnico que probablemente que aparece poco en el dataset de fine-tuning.

**Análisis de pares control**

6 Tabla 2.5 Comparación semántica Modelo Generalista BGE-M3 vs. Versión fine-tuneada en Pares de control

| **Par de términos** | **Similitud Base** | **Similitud Fine-Tune** | **Cambio Absoluto** | **Cambio Relativo** |
| --- | --- | --- | --- | --- |
| cáncer colorrectal ↔ infarto | 0.453 | 0.257 | -0.196 | -43.3% |
| cáncer ↔ asma | 0.418 | 0.104 | -0.314 | -75.0% |
| recidiva del cáncer ↔ alergia alimentaria | 0.438 | 0.085 | -0.353 | -80.5% |
| ictus ↔ hipoglucemia | 0.497 | 0.269 | -0.229 | -46.0% |
| edema ↔ fiebre alta | 0.466 | 0.263 | -0.203 | -43.5% |
| tumor de pulmón ↔ gripe | 0.458 | 0.241 | -0.217 | -47.4% |
| depresión ↔ hipertensión | 0.567 | 0.234 | -0.333 | -58.7% |
| epilepsia ↔ alergia | 0.642 | 0.338 | -0.304 | -47.3% |

En todos los casos, la similitud disminuye, como es deseable para pares sin relación semántica. Esto indica que el modelo no ha introducido asociaciones falsas tras el fine-tuning.

## Pipeline de indexación y almacenamiento vectorial en ChromaDB

## Selección de ChromaDB como sistema de almacenamiento vectorial

ChromaDB fue seleccionado frente a alternativas como Pinecone, Weaviate o FAISS por las siguientes ventajas:

* ChromaDB proporciona una solución integral que combina el almacenamiento de embeddings vectoriales con metadatos estructurados en una misma base de datos. Esta característica elimina la complejidad de mantener sistemas separados para vectores y metadatos
* **Compatibilidad nativa con el ecosistema de embeddings**: La integración directa con SentenceTransformers facilita el despliegue de modelos de embeddings como BGE-M3
* **Capacidades de filtrado especializado**: El sistema permite aplicar filtros granulares por categorías médicas específicas durante la fase de búsqueda vectorial.
* Se ejecuta localmente garantizando reproducibilidad.
* Como solución open-source, ChromaDB garantiza la transparencia total del código fuente, elimina las dependencias de licenciamiento comercial.

## Arquitectura del Pipeline de indexación

El proceso de indexación se implementa a través del script **1-init\_indexing.py**, que coordina un pipeline modular diseñado específicamente para documentos médicos heterogéneos, integrando extracción, limpieza, fragmentación, enriquecimiento semántico e indexación en ChromaDB.

## Configuración del sistema

La parametrización del sistema se gestiona mediante el archivo **config.yaml**, permitiendo ajustar componentes clave:

Entre sus parámetros clave se encuentran:

* **Modelos de embedding**: BAAI/bge-m3 (multilingüe generalista) y un modelo fine-tuneado bge\_m3\_epochs/epoch4\_MRR0.9717.
* **Modelo biomédico español**: BSC-LT/roberta-base-biomedical-es, utilizado para la extracción de palabras clave.
* **Segmentación**: chunks de 400 caracteres con un solapamiento de 80 caracteres.
* **Activación del enriquecimiento semántico** con KeyBERT.

***collection****:*

*name: "documentos\_osakidetza"*

*name\_finetuneado: "documentos\_finetuneado\_bge\_m3"*

***model****:*

*name\_embedding: "BAAI/bge-m3"*

*name\_finetuning: "bge\_m3\_epochs/epoch4\_MRR0.9717"*

*name\_es: "BSC-LT/roberta-base-biomedical-es"*

***preprocessing****:*

*chunk\_size: 400*

*chunk\_overlap: 80*

*enrich\_keywords: true*

## Procesamiento multi-fuente

* **Extracción de PDFs**: Se utiliza la función extract\_pdf\_text () anteriormente explicada y utilizada también para la creación del dataset\_test.json.
* **Scraping Web**: Se aplica la función extract\_web\_text() anteriormente explicada y utilizada también para la creación del dataset\_test.json.

## Segmentación (Chunking)

La fragmentación emplea RecursiveCharacterTextSplitter con parámetros optimizados para preservar la coherencia clínica:

* **Tamaño de chunk**: 300 caracteres para maximizar el contexto médico preservado
* **Solapamiento**: 50 caracteres para garantizar continuidad semántica

## Enriquecimiento semántico con KeyBERT

Para maximizar la precisión en la recuperación, cada chunk se complementa con 5 palabras clave extraídas mediante:

* **Modelo base**: BSC-LT/roberta-base-biomedical-es, preentrenado en texto biomédico en español.
* **Algoritmo MMR** (Maximal Marginal Relevance), con un parámetro de diversidad de 0.2.
* **n-gramas** de 1 y 2 palabras, para capturar tanto términos clínicos como conceptos generales relacionados.

## Función personalizada de embeddings

Para optimizar la generación de representaciones vectoriales, se desarrolló la clase **FuncionEmbeddingsPersonalizada**, una implementación técnica especializada que actúa como interfaz entre los modelos de transformers y ChromaDB.

La función ejecuta un proceso optimizado de conversión texto-vector:

1. **Inicialización del modelo**: Carga un modelo de transformers (como BGE-M3) junto con su tokenizer correspondiente, configurando las capacidades de procesamiento según la arquitectura específica.
2. **Tokenización**: Cuando recibe una lista de textos, los convierte a secuencias numéricas que el modelo puede procesar, aplicando padding y truncamiento según sea necesario para mantener longitudes consistentes.
3. **Inferencia sin entrenamiento**: Pasa los tokens tokenizados a través del modelo transformer en modo evaluación (sin actualizar pesos), extrayendo las representaciones contextuales de cada token.
4. **Pooling adaptativo**: Aplica la estrategia de pooling óptima según la arquitectura:

* CLS pooling en modelos BGE y RoBERTa: extrae directamente el vector del token [CLS]
* Mean pooling con máscara de atención en BERT: calcula el promedio ponderado de todos los vectores de tokens, excluyendo posiciones de padding mediante la máscara de atención. La máscara de atención evita que tokens de padding contaminen la representación final.

1. **Normalización L2:** Normaliza todos los vectores resultantes, garantizando que las búsquedas en ChromaDB utilicen distancia coseno (similaridad angular) en lugar de distancia euclidiana, eliminando el sesgo por longitud vectorial.
2. **Entrega optimizada**: Devuelve la lista de vectores normalizados, listos para indexación en ChromaDB.

## Implementación de colecciones vectoriales diferenciadas

Con el objetivo de analizar empíricamente el impacto del fine-tuning en tareas de recuperación médica, se han implementado dos colecciones vectoriales diferenciadas en ChromaDB:

* **Colección generalista**: construida a partir del modelo multilingüe BAAI/bge-m3 mediante una función de embedding personalizada.
* **Colección especializada**: basada en un modelo fine-tuneado (bge\_m3\_epochs/epoch4\_MRR0.9717) y procesada mediante la misma función de embedding.

## Estructura de almacenamiento en ChromaDB

Cada fragmento se almacena como una unidad semánticamente enriquecida que incluye:

* **Identificador único (UUID)**
* **Contenido textual limpio y segmentado**
* **Metadatos estructurados** que facilitan la recuperación granular y la trazabilidad completa

**Ejemplo de entrada en ChromaDB:**

{

"id": "251e2948-c6c2-4e98-911d-eef5a85a3521",

"**document\_id**": "pdf\_consejos\_gripe\_general\_es",

"**chunk\_id**": "pdf\_consejos\_gripe\_general\_es\_chunk0",

"**chunk\_position**": "1/7",

"**filename"**: "consejos\_gripe\_general\_es.pdf",

"**fuente**": "PDF",

"**idioma**": "es",

"**publico\_objetivo**": "general",

"**categoria**": "enfermedades\_comunes",

"**subcategoria**": "Gripe",

"**length\_chars**": 282,

"**length\_tokens**": 49,

"**next\_chunk**": "pdf\_consejos\_gripe\_general\_es\_chunk1",

"**palabras\_clave**": "gripe, fiebre alta, virus, escalofrios",

"**chunk\_keywords**": "osasun saila departamento, se expulsan, salud gripe se contagia"

}

**Contenido textual asociado:**

*"Algunas veces puede ocasionar complicaciones (sinusitis, otitis...) o hacer que se agraven algunas enfermedades crónicas. Es una enfermedad, producida por un virus, más frecuente en otoño e invierno. Suele iniciarse de forma brusca, con fiebre alta (más de 38º C). Además, suele ir acompañada de escalofríos, malestar general, dolor de cabeza, dolores musculares y articulares, lagrimeo, tos."*

El diseño del esquema de metadatos responde a criterios de optimización semántica y funcional:

* **Trazabilidad jerárquica**: los campos document\_id, chunk\_id y chunk\_position permiten reconstruir el contexto completo de un documento desde sus fragmentos individuales.
* **Clasificación médica**: categoria y subcategoria habilitan filtros temáticos.
* **Keywords multinivel**: se almacenan tanto palabras clave del documento completo (palabras\_clave) como específicas de cada chunk (chunk\_keywords), lo que favorece una recuperación híbrida léxico-semántica.
* **Contexto estructural:** atributos como next\_chunk, length\_chars y length\_tokens posibilitan estrategias de recuperación contextual y generación de respuestas extendidas.
* **Metadatos técnicos:** campos como fuente, idioma y publico\_objetivo permiten personalizar el sistema según el perfil del usuario final.

## Validación y monitorización del corpus indexado

El sistema integra funciones de validación que generan métricas detalladas para evaluar la calidad del corpus:

7 Tabla 2.6 Estadísticas del proceso de chunking

|  |  |
| --- | --- |
| *"n\_chunks":* | *8304* |
| *"chars\_avg":* | *226.84* |
| *"chars\_min":* | *26* |
| *"chars\_max":* | *300* |
| *"chars\_p10"* | *101.0* |
| *"chars\_p90”* | *298.0* |
| *“tokens\_avg”* | *33.75,* |
| *"tokens\_min":* | *5* |
| *"tokens\_max":* | *58* |

Análisis de la calidad del corpus:

* **Fragmentación equilibrada**: media de 226.84 caracteres por chunk, con una dispersión controlada (P10–P90).
* **Alta densidad semántica**: Con un promedio de 33.75 tokens por chunk y un rango de 5-58 tokens, los os fragmentos mantienen suficiente contexto sin redundancia.
* **Rango**: Con longitud mínima de 26 y máxima de 300 caracteres asegura que todos los fragmentos contienen información sustancial.
* **Distribución equilibrada**: Los percentiles P10 (101 caracteres) y P90 (298 caracteres) demuestran una distribución concentrada que evita fragmentos demasiado pequeños o excesivamente largos.
* **Métricas de Validación Técnica**
* **Cobertura completa**: Procesamiento exitoso del 100% de documentos fuente
* **Integridad de metadatos**: Enriquecimiento completo con identificadores únicos, clasificación médica y keywords especializadas
* **Trazabilidad garantizada**: Cada chunk mantiene vínculos bidireccionales con su documento origen y fragmentos adyacentes

## Validación Semántica con UMAP

Para comprobar la coherencia semántica de los embeddings generados, se aplicó la técnica de reducción de dimensionalidad UMAP (Uniform Manifold Approximation and Projection), proyectando los vectores de alta dimensión (1024D) a un espacio bidimensional (2D).

Se utilizó el script **visualizar\_umap.py**, el cual extrae embeddings y metadatos directamente desde la colección ChromaDB, aplica la transformación UMAP con parámetros optimizados para datos textuales médicos, y genera visualizaciones categorizadas.

Las figuras muestran la proyección bidimensional del espacio semántico de los chunks indexados mediante UMAP, una técnica de reducción de dimensionalidad que preserva la estructura local del espacio vectorial de alta dimensión. Cada punto representa un fragmento textual (chunk), coloreado según su categoría médica.

**Objetivo del Análisis**

Evaluar visualmente si el espacio de embeddings generado (utilizando el modelo BAAI/bge-m3 o su versión fine-tuneada) refleja una separación semántica coherente entre distintas categorías médicas. Esto permite validar si los embeddings capturan adecuadamente relaciones de similitud semántica relevantes para la recuperación de información.

Mapa, Gráfico de dispersión

El contenido generado por IA puede ser incorrecto.

3. 2.3 Validación UMAP del Modelo Base BAAI/bge-m3: clustering semántico por categorías médicas "

**Modelo Base BAAI/bge-m3**

* **Salud mental (cuadrados azules)** ocupa una zona muy amplia y densa en la parte inferior. Aunque hay algo de dispersión, la mayoría de puntos están bien agrupados, lo que indica que el modelo ha identificado patrones semánticos comunes dentro de esta categoría**.**
* **Cáncer (rombos verdes) y Cardiovascular (rombos amarillos)** también muestran agrupaciones bastante compactas. Es un buen resultado, ya que son temas clínicos con vocabulario especializado, y el modelo los ha sabido diferenciar bien.
* **Respiratorias (cruces rosas)** aparecen mezcladas en diferentes zonas. Esto puede deberse a que comparten síntomas o tratamientos con otras categorías, como salud mental o enfermedades comunes.
* **ITS (triángulos beige) y neurología (cruces grises)** aparecen en zonas bastante separadas del resto, lo cual creo que es una buena señal de que el modelo ha captado su especificidad.
* En el centro del gráfico hay una zona donde se solapan varias categorías. Se puede llegar a la conclusión que aquí están los fragmentos más genéricos o que tratan temas comunes como síntomas o cuidados generales, lo cual puede explicar la superposición.

Gráfico, Mapa, Gráfico de dispersión

El contenido generado por IA puede ser incorrecto.

4- 2.4 Proyección UMAP de embeddings generados por el modelo fine-tuneado (epoch4\_MRR0.9717). clustering semántico por categorías médicas

**Modelo Fine-tuneado**

* **Salud mental (cuadrados azul oscuro)** forma un conglomerado denso y bien definido en la parte inferior derecha, lo que indica que los embeddings agrupan bien los temas que comparten una terminología específica..
* **Cáncer (rombos verdes) y Cardiovascular (rombos amarillos)** también presentan regiones diferenciadas con alta densidad, aunque con algunas zonas de intersección, lo que sugiere conceptos compartidos o ambigüedad léxica en algunos casos (ej., "fatiga", "dolor", "tumor").
* **Enfermedades comunes y respiratorias están más mezcladas**: Estas categorías aparecen más dispersas o entrelazadas con otras. Se intuye que es porque usan vocabulario más general o comparten síntomas con otras enfermedades, lo cual puede dificultar que el modelo los separe perfectamente.
* **Respiratorias y diabetes**, aunque más dispersas, mantienen agrupaciones locales, lo que apunta a cierta coherencia semántica, pero también a una posible superposición de términos comunes con otras patologías.
* **ITS - Enfermedades de Transmisión Sexual (triángulos beige) y neurología (cruces grises)** se ubican en regiones más aisladas, lo cual es positivo: su vocabulario técnico es distintivo y los embeddings lo capturan bien.
* Se observan puntos de intersección entre enfermedades comunes, cáncer, cardiovascular y salud mental, lo cual es esperable dada la coexistencia de síntomas comunes o interacciones clínicas en el discurso médico.
* La zona central de la proyección actúa como región de "transición semántica", donde se mezclan categorías, posiblemente por contenido genérico o no específico.
* Cuando dos categorías aparecen mezcladas (como respiratorias y enfermedades comunes), puede ser útil añadir metadatos o aplicar un reranking posterior para afinar más la respuesta.

8 Tabla 2.7 Comparativa: Modelo generalista vs. Modelo fine-tuneado

| **Criterio** | **Modelo BAAI/bge-m3 (generalista)** | **Modelo fine-tuneado (epoch4\_MRR0.9717)** |
| --- | --- | --- |
| Agrupamiento semántico | Correcto en categorías grandes, pero más difuso en zonas mixtas | Más compacto, mejor separación entre categorías |
| Ruido en zonas centrales | Alto solapamiento entre categorías (cáncer, respiratorias, comunes) | Menor solapamiento y fronteras más claras |
| Separación de categorías pequeñas (ITS, neurología) | Aisladas pero algo dispersas | Bien agrupadas y más cohesivas |
| Salud mental | Amplia pero ligeramente dispersa | Más cohesionada y con límites más definidos |
| Capacidad de generalización | Alta: reconoce múltiples dominios, útil para documentos variados | Focalizada en lenguaje clínico: más eficaz si los documentos son del mismo dominio |
| Ventaja práctica en recuperación médica | Buena base inicial, pero puede fallar en preguntas clínicas específicas | Mejor precisión en temas médicos complejos o específicos |

# Metodología experimental

## Introducción

El presente capítulo tiene como objetivo principal describir la metodología experimental empleada para la evaluación comparativa de distintas estrategias de recuperación de información médica. Dado el carácter crítico del acceso a información fiable en el ámbito clínico, resulta esencial analizar el rendimiento de diversas técnicas —desde enfoques tradicionales basados en representaciones dispersas como **TF-IDF** **o BM25**, hasta modelos modernos que utilizan **embeddings densos** y **reranking neuronal**—.

La evaluación se basa en la aplicación de métricas estándar de recuperación de información como Precision@K, Recall@K, F1@K, MRR y nDCG, que permiten medir la precisión, cobertura y calidad del ranking generado por cada método de búsqueda.

Para ello, se utiliza un conjunto de datos de test implementado en el archivo dataset\_test.json y descrito previamente en el apartado **2.1 Creación del dataset de evaluación**. Este conjunto fue generado a partir de consultas clínicas realistas, elaboradas con ayuda de modelos generativos y validadas manualmente, ya que las preguntas no presentaban una complejidad clínica elevada.

Esta validación permite establecer con claridad qué documentos son relevantes para cada consulta, sirviendo como ground truth para el cálculo de métricas objetivas.

## Marco metodológico general

La metodología experimental se ha diseñado para garantizar la **validez, reproducibilidad y comparabilidad** de los resultados obtenidos, basándose en los siguientes principios:

1. **Reproducibilidad**

Todos los experimentos se ejecutan mediante scripts parametrizables definidos en archivos config.yaml, lo que permite repetir exactamente las mismas condiciones de evaluación en cualquier momento o entorno.

1. **Comparabilidad**

Se asegura una comparación justa entre métodos manteniendo constantes el corpus documental, las queries y las métricas evaluadas. Esto significa que cualquier diferencia observada en el rendimiento es atribuible únicamente al método de recuperación utilizado.

1. **Trazabilidad**

Cada experimento genera resultados estructurados en archivos .xlsx que incluyen tanto métricas agregadas por método como resultados detallados por query. Esta estructura permite realizar análisis cuantitativos (por ejemplo, precisión promedio) y cualitativos (por ejemplo, revisión de los documentos recuperados en cada consulta).

1. **Modularidad y escalabilidad**

El código experimental sigue una arquitectura modular, lo que permite integrar fácilmente nuevas técnicas, modificar componentes del pipeline (por ejemplo, cambiar el modelo de embeddings o el sistema de indexación) y escalar la evaluación a conjuntos de datos más grandes o diferentes dominios médicos.

## Arquitectura experimental

La arquitectura experimental ha sido diseñada para garantizar un entorno de evaluación robusto, automatizado y fácilmente replicable. Este diseño facilita tanto la ejecución de múltiples experimentos como el análisis comparativo entre distintas estrategias de recuperación en el dominio clínico.

## Infraestructura

Los experimentos se han implementado en forma de **scripts modulares** organizados en carpetas específicas (retrieval/, evaluation/), con todos sus parámetros controlados mediante archivos de configuración **config.yaml**. Este enfoque permite ejecutar experimentos simplemente modificando las variables de configuración, sin necesidad de alterar el código principal.

Cada script está instrumentado con logging detallado para garantizar la trazabilidad completa de cada experimento: desde el modelo de embeddings utilizado, hasta los valores de K evaluados, los tiempos de ejecución y las métricas generadas. Además, se emplean semillas aleatorias fijas para asegurar resultados deterministas.

Esta infraestructura está pensada para escalar fácilmente, permitiendo ejecutar los mismos experimentos sobre distintos modelos, configuraciones o corpus, con mínima intervención manual.

## Estructura de los resultados

Cada experimento produce dos tipos de archivos de resultados en formato Excel (.xlsx), que permiten un análisis detallado y ordenado:

* **\*\_resumen.xlsx**: Contiene las métricas agregadas por cada método de recuperación y valor de K. Las columnas incluyen method, k, precision, recall, f1, mrr, ndcg. Este archivo permite análisis estadísticos rápidos y comparaciones globales entre métodos.
* **\*\_detalles.xlsx**: Registra los resultados individuales por query, incluyendo además información adicional sobre los chunks recuperados:
  + query: la consulta evaluada
  + topk: lista de documentos o chunks recuperados
  + precision, recall, f1, mrr, ndcg: métricas por consulta
  + chunk\_id, chunk\_position, chunk\_text: metadatos que permiten rastrear exactamente qué fragmento fue recuperado, en qué posición del documento estaba, y con qué score.

## Métricas de evaluación

La evaluación de los distintos métodos de recuperación se ha realizado mediante un conjunto de métricas estándar ampliamente utilizadas en el campo de la recuperación de información, especialmente en sistemas RAG (Retrieval-Augmented Generation).

La selección de estas métricas ha sido guiada por las recomendaciones prácticas recogidas en fuentes como Yapi Kredi Teknoloji (2024), y permite analizar de forma integral la calidad, relevancia y orden de los documentos recuperados.

Las métricas han sido calculadas para diferentes valores de K (1, 3, 5 y 10), con el fin de evaluar el rendimiento en distintos escenarios de recuperación. K refleja el número de documentos a considerar en el ranking para evaluar las métricas.

## Métricas principales

* **Precision@K**: mide qué proporción de los *K* documentos recuperados son realmente relevantes.
* **Recall@K:** mide el porcentaje de documentos relevantes se encontraron en el top-K
* **F1@K**: midel la media armónica entre precisión y recall.
* **MRR (Mean Reciprocal Rank):** indica en qué posición aparece el primer documento relevante. Cuanto más cerca esté del inicio, mejor.
* **nDCG@K (Normalized Discounted Cumulative Gain):** evalúa el orden de los documentos recuperados, ponderando la relevancia por su posición en el ranking.

## Métricas auxiliares

Durante la evaluación se calculan también métricas auxiliares por cada *query*, útiles para análisis más cualitativos:

* **TP (True Positives)**: documentos recuperados y presentes en el *ground truth*.
* **FP (False Positives)**: documentos recuperados pero no relevantes.
* **FN (False Negatives)**: documentos relevantes no recuperados.

Estas métricas se han calculado utilizando el dataset dataset\_test.json, descrito en el apartado 2.1, que contiene preguntas manuales evaluadas sobre un corpus médico controlado.

## Dataset de evaluación

Para la evaluación de los métodos de recuperación se ha utilizado el dataset dataset\_test.json, descrito detalladamente en el apartado **2.1 Creación del dataset de evaluación**. Este conjunto de pruebas está compuesto por 322 queries médicas manuales, distribuidas en 8 categorías clínicas y 18 subcategorías específicas, cubriendo un amplio espectro temático del dominio sanitario.

El dataset incluye para cada pregunta un conjunto de documentos relevantes definidos manualmente, lo que permite calcular métricas de recuperación de manera precisa. Esta estructura garantiza la validez del experimento y permite realizar comparaciones controladas entre distintos enfoques de recuperación.

## Diseño experimental por métodos

Con el fin de garantizar una comparación justa entre las distintas técnicas de recuperación de información, todos los experimentos se han diseñado bajo condiciones estrictamente controladas. Cada método ha sido evaluado sobre el mismo conjunto de consultas, con los mismos documentos fuente y las mismas métricas de evaluación, asegurando así la validez de los resultados comparativos.

## Condiciones controladas

* **Corpus común**: Todos los métodos se han evaluado sobre el mismo conjunto de documentos.
* **Consultas idénticas**: Se ha utilizado el mismo conjunto de 322 preguntas para cada experimento.
* **Ground truth compartido**: Las anotaciones de documentos relevantes han sido constantes en todas las pruebas.
* **Métricas homogéneas**: Precision@K, Recall@K, F1@K, MRR y nDCG, evaluadas para K ∈ {1, 3, 5, 10}.
* **Chunking coherente**: Para experimentos que operan a nivel de fragmento, se mantiene una segmentación uniforme del contenido.

## Implementación modular

Los experimentos han sido organizados mediante scripts autónomos y configurables, siguiendo una arquitectura modular basada en:

* Automatización por config.yaml: Cada script permite parametrizar modelos, corpus, métricas y rutas de salida.
* Separación de responsabilidades: Los pasos de carga de datos, inferencia, evaluación y visualización están desacoplados para facilitar su mantenimiento.
* Registro completo (logging): Cada ejecución genera trazas detalladas que permiten auditar y depurar el proceso experimental.

## Modelos evaluados y configuración

Este apartado describe los modelos utilizados para los experimentos de recuperación semántica, diferenciando entre un modelo generalista y otro especializado mediante fine-tuning en el dominio médico. Se detallan además los aspectos técnicos de la configuración para asegurar reproducibilidad y eficiencia computacional.

* **Modelo generalista: BAAI/bge-m3:**

Se empleó como punto de partida el modelo BAAI/bge-m3, un encoder multilingüe preentrenado sobre tareas de búsqueda semántica, recuperación de información y clasificación. Este modelo se caracteriza por:

* + Arquitectura basada en transformers optimizados para tareas de dense retrieval.
  + Soporte multilingüe, lo que lo hace especialmente útil para trabajar en contextos como el sistema sanitario español.
  + Capacidad de funcionamiento tanto como Bi-Encoder (consulta y documento embebidos por separado) como Cross-Encoder (pares evaluados conjuntamente), lo que permite comparaciones metodológicas dentro de una misma arquitectura.

La elección de este modelo se debe a su balance entre rendimiento y eficiencia computacional, además de su amplia aceptación como baseline fuerte en tareas RAG.

* **Modelo fine-tuneado: bge\_m3\_epochs/epoch4\_MRR0.9717**

Sobre el modelo generalista se aplicó un proceso de fine-tuning específico para el dominio médico, utilizando un conjunto de 2.033 pares pregunta-documento generados a partir del corpus y validados manualmente. El modelo ajustado presenta las siguientes características:

* + Entrenado durante varias épocas con optimización supervisada en función de la similaridad semántica.
  + El checkpoint seleccionado (epoch 4) alcanzó un MRR (Mean Reciprocal Rank) de 0.9717 sobre el conjunto de validación, demostrando una alta capacidad para capturar relaciones médico-clínicas.

## Experimento 1: TF-IDF + Bi-Encoder + Cross-Encoders

Este experimento, implementado en el script **1-Tfidf-CrossEncoder\_biencoder.py**, desarrolla un pipeline híbrido de tres etapas que combina distintas técnicas de recuperación para evaluar sus ventajas relativas en el dominio médico.

## Objetivos e hipótesis

**Objetivo**: Evaluar comparativamente distintas estrategias de recuperación de información médica a nivel de documento completo, combinando métodos clásicos basados en representaciones dispersas (TF-IDF) con métodos densos (Bi-Encoder) y técnicas de reranking (Cross-Encoder).

**Objetivos específicos**:

* **Comparación metodológica controlada**: Analizar el desempeño individual de cada técnica evaluando el mismo contenido documental completo, asegurando que las diferencias observadas se atribuyan exclusivamente a la capacidad de la técnica de recuperación.
* **Evaluación de eficiencia computacional**: Determinar si el uso de un pool balanceado —formado por los candidatos mejor valorados por TF-IDF y Bi-Encoder— optimiza el rendimiento del Cross-Encoder sin comprometer la precisión ni aumentar significativamente el coste computacional.

**Hipótesis de trabajo**:

1. **Superioridad semántica**: Los métodos densos (Bi-Encoder) y técnicas de reranking (Cross-Encoder)superan a TF-IDF en su capacidad para recuperar documentos clínicamente relevantes, gracias a su habilidad para identificar similitudes semánticas más allá de coincidencias léxicas.
2. **Eficiencia del pool balanceado**: La combinación paralela de candidatos provenientes de TF-IDF y Bi-Encoder en la etapa de reranking mejora la diversidad y relevancia de los resultados sin necesidad de evaluaciones exhaustivas.
3. **Impacto del fine-tuning clínico**: El modelo ajustado específicamente al dominio médico (bge\_m3\_epochs/epoch4\_MRR0.9717) proporciona mejoras sustanciales en métricas clave (MRR, nDCG) respecto a su contraparte generalista (BAAI/bge-m3), especialmente en las fases densas del pipeline.

**Justificación: ´**

La recuperación de información en el dominio médico requiere una alta sensibilidad semántica para identificar conceptos clínicos expresados de múltiples formas.

Las representaciones dispersas como TF-IDF, aunque eficientes, se basan en coincidencias léxicas sin comprender el significado subyacente.

En cambio, los métodos densos, como los Bi-Encoders y especialmente los Cross-Encoders, proyectan texto en espacios vectoriales donde la cercanía refleja similitud semántica, y permiten evaluar directamente el grado de relación entre una consulta y un documento de forma contextualizada.

Evaluar estos métodos bajo condiciones controladas —mismo corpus, mismas consultas— y utilizando métricas estándar como Precision@k, Recall, MRR y nDCG, garantiza una comparación justa y útil para tomar decisiones informadas en el diseño del sistema RAG posterior.

## Metodología específica

**Etapa 1**: Recuperación con TF-IDF

* **Descripción**: TF-IDF transforma cada documento del corpus y cada consulta médica en vectores dispersos que reflejan la frecuencia y especificidad de los términos. Luego se calcula la similitud coseno para obtener un ranking de documentos más relevantes.
* **Preparación del corpus:** Se extraen los textos de cada documento y se genera la matriz TF-IDF:

*doc\_ids = list(corpus.keys())*

*docs = [corpus[doc\_id] for doc\_id in doc\_ids]*

*X = vectorizer.fit\_transform(docs)*

* **Parámetros utilizados para configurar y entrenar el vectorizador:**

Se genera una matriz de [45\_documentos × 15000\_características]

*# Configurar y entrenar TF-IDF*

*vectorizer = TfidfVectorizer(*

*ngram\_range=(1, 2),  # Unigrams y bigrams*

*max\_features=15000,  # Máximo 15k características*

*stop\_words=None,     # No remover stop words (importantes en medicina)*

*lowercase=True,      # Convertir a minúsculas*

*strip\_accents='unicode',  # Normalizar acentos*

*min\_df=2,         ## Términos en al menos 2 documentos*

*max\_df=0.95       # Términos en máximo 95% documentos*

*)*

*X = vectorizer.fit\_transform(docs)*

* **Para cada query individual:**
  + La consulta se transforma en un vector de características TF-IDF utilizando el vocabulario pre-entrenado sobre el corpus completo
  + Se calcula la similitud coseno entre el vector de la consulta y cada uno de los 45 documentos del corpus. **TfidfVectorizer de scikit-learn** por defecto aplica **normalización L2**, que significa que cada vector ya tiene norma = 1, así que cuando ambos vectores están normalizados (norma = 1), el producto punto es la similitud coseno.
  + Los documentos se ordenan por similitud descendente, generando un ranking completo:

*tfidf\_ranking =* ***calculate\_tfidf\_rankings****(vectorizer, X, doc\_ids, query\_text)*

***def calculate\_tfidf\_rankings****(vectorizer, X, doc\_ids, query\_text):*

*"""TF-IDF - Búsqueda  por palabras clave"""*

*logger.debug(f"TF-IDF: Procesando query '{query\_text[:50]}...'")*

*qv = vectorizer.transform([query\_text])*

*sims = (X @ qv.T).toarray().ravel()*

*idx = sims.argsort()[::-1]*

*return [doc\_ids[i] for i in idx]*

**Etapa 2:** Reordenación Semántica con Bi-Encoder

* **Descripción:** El Bi-Encoder transforma tanto la consulta como los documentos en vectores densos dentro de un espacio semántico compartido. Se calcula la similitud coseno entre la consulta y cada documento. Este enfoque permite detectar relaciones semánticas más allá de coincidencias léxicas.

La consulta se codifica una vez generando un embedding de 1024 dimensiones.

Los 45 documentos del corpus se procesan en lotes de 16 para optimizar el uso de memoria GPU, generando embeddings del mismo tamaño.

La similitud coseno se calcula mediante el producto punto normalizado entre el vector de la consulta y cada vector documental, produciendo scores de relevancia que permiten ordenar todos los documentos de mayor a menor similitud semántica.

* **Modelos utilizados:**
  + Modelo generalista multilingüe: BAAI/bge-m3
  + Modelo fine-tuneado clínico: bge\_m3\_epochs/epoch4\_MRR0.9717

***if config['model']['use\_finetuned']:***

*model\_dir = config['paths']['model\_path']*

*model\_name = config['model']['name\_finetuning']*

*full\_model\_path = os.path.join(model\_dir, model\_name)*

***biencoder = SentenceTransformer(full\_model\_path)***

*else:*

***biencoder = SentenceTransformer(config['model']['name\_embedding'])***

* **Preparación de datos:**

Se parte del mismo conjunto de documentos que en la etapa anterior.

Primero se extraen los IDs y los textos del corpus:

*doc\_ids = list(corpus.keys())*

*docs = [corpus[doc\_id] for doc\_id in doc\_ids]*

* **Cálculo del ranking con Bi-Encoder (procesamiento por lotes):**

El Bi-Encoder transforma la consulta y todos los documentos en vectores densos, luego calcula la similitud coseno para obtener el ranking final.

La consulta se codifica una vez generando un embedding de 1024 dimensiones. Los 45 documentos del corpus se procesan en lotes de 16 para optimizar el uso de memoria GPU, generando embeddings del mismo tamaño.

La similitud coseno se calcula mediante el producto punto normalizado entre el vector de la consulta y cada vector documental, produciendo scores de relevancia que permiten ordenar todos los documentos de mayor a menor similitud semántica. La similitud se calcula por lote

Procesar todos los documentos juntos en GPU causa errores de memoria, por lo que se opta por un procesamiento por lotes

def **calculate\_biencoder\_rankings**(biencoder, corpus, doc\_ids, query\_text, device, batch\_size=16):

    """Bi-Encoder con procesamiento por batches para evitar OOM"""

logger.debug(f"Bi-Encoder: Procesando query '{query\_text[:50]}...'")

q\_emb = biencoder.encode(query\_text, convert\_to\_tensor=True, device=device)

docs\_texts = [corpus[doc\_id] for doc\_id in doc\_ids]

# Procesar documentos en batches

all\_similarities = []

for i in range(0, len(docs\_texts), batch\_size):

batch\_texts = docs\_texts[i:i+batch\_size]

batch\_embs = biencoder.encode(batch\_texts, convert\_to\_tensor=True, device=device)

# Calcular similitudes para este batch

batch\_similarities = []

**for doc\_emb in batch\_embs**:

**sim = float(torch.cosine\_similarity(q\_emb.unsqueeze(0), doc\_emb.unsqueeze(0)))**

**batch\_similarities.append(sim**)

all\_similarities.extend(batch\_similarities)

# Libera memoria GPU después de cada lote

del batch\_embs

torch.cuda.empty\_cache()

    # Ranking final. Similitudes + IDs de documentos

ranked\_pairs = sorted(zip(all\_similarities, doc\_ids), key=lambda x: x[0], reverse=True)

# **Devuelve**: Lista ordenada de document IDs

return [doc\_id for \_, doc\_id in ranked\_pairs]

* **Uso en el pipeline:**

biencoder\_ranking = calculate\_biencoder\_rankings(biencoder, corpus, doc\_ids, query\_text, device, batch\_size=biencoder\_batch\_size)

**Etapa 3: Reordenación final con Cross-Encoder**

* **Descripción:** En esta fase se aplica un Cross-Encoder que evalúa directamente cada par (consulta, documento) introduciendo ambos textos simultáneamente en el modelo. Esto permite que el modelo capte relaciones de contexto cruzado palabra a palabra. Esta representación está contextualizada, ya que considera cómo interactúan las palabras en conjunto, lo cual incrementa significativamente la precisión semántica.
* **Objetivo:** Aplicar una evaluación más precisa mediante reranking contextualizado, maximizando la relevancia semántica.

Aunque esta etapa es la más costosa computacionalmente, su precisión es superior.

* **Modelo utilizado:** BAAI/bge-reranker-v2-m3, complementario al Bi-Encoder BAAI/bge-m3.
* **Implementación técnica:**
  + **Construcción del pool balanceado:** Para asegurar imparcialidad, el modelo opera sobre un pool balanceado de documentos seleccionados tanto por TF-IDF (sparse) como por Bi-Encoder (denso), afinando el ranking final a partir de esa diversidad metodológica.

En lugar de tomar solo los top documentos de uno u otro método, se forma un pool mixto balanceado combinando los resultados de ambas estrategias, de tal forma que se consigue una selección equitativa de candidatos

*balanced\_pool = create\_balanced\_pool(tfidf\_ranking, biencoder\_ranking, pool\_size \* 5)*

Esto asegura que el Cross-Encoder evalúe una variedad más amplia de documentos, reduciendo el riesgo de limitarse a un solo tipo de recuperación.

La función **create\_balanced\_pool(...)** combina equitativamente los mejores documentos de cada método anterior:

*def create\_balanced\_pool(tfidf\_ranking, biencoder\_ranking, pool\_size=50):*

*"""Pool Balanceado - Eliminando sesgos metodológicos"""*

*logger.debug(f"Creando pool balanceado de {pool\_size} documentos...")*

*half\_size = pool\_size // 2*

*tfidf\_pool = tfidf\_ranking[:half\_size]*

*biencoder\_pool = biencoder\_ranking[:half\_size]*

*seen = set()*

*balanced\_pool = []*

*max\_iterations = max(len(tfidf\_pool), len(biencoder\_pool))*

*for i in range(max\_iterations):*

*if i < len(tfidf\_pool) and tfidf\_pool[i] not in seen:*

*balanced\_pool.append(tfidf\_pool[i])*

*seen.add(tfidf\_pool[i])*

*if i < len(biencoder\_pool) and biencoder\_pool[i] not in seen:*

*balanced\_pool.append(biencoder\_pool[i])*

*seen.add(biencoder\_pool[i])*

*if len(balanced\_pool) >= pool\_size:*

*break*

*logger.debug(f"Pool balanceado creado: {len(balanced\_pool)} documentos únicos")*

*return balanced\_pool*

* + **Reranking con Cross-Encoder:**

Este conjunto balanceado de documentos candidatos se envía al Cross-Encoder para evaluación final:

crossencoder\_ranking = **calculate\_crossencoder\_rankings**(cross\_encoder, corpus, balanced\_pool, query\_text, batch\_size)

def **calculate\_crossencoder\_rankings**(cross\_encoder, corpus, pool\_docs, query\_text, batch\_size=8):

    """Cross-Encoder -"""

    if not pool\_docs:

        logger.debug("Cross-Encoder: Pool vacío, devolviendo lista vacía")

        return []

    logger.debug(f"Cross-Encoder: Evaluando {len(pool\_docs)} documentos candidatos")

    pairs = []

    for doc\_id in pool\_docs:

        document\_text = corpus[doc\_id]

        pair = (query\_text, document\_text)

        pairs.append(pair)

    scores = cross\_encoder.predict(pairs, batch\_size=batch\_size)

    ranked\_pairs = sorted(zip(scores, pool\_docs), key=lambda x: x[0], reverse=True)

    return [doc\_id for \_, doc\_id in ranked\_pairs]

* **Uso en el pipeline:**

biencoder\_ranking, pool\_size=10)

crossencoder\_ranking = calculate\_crossencoder\_rankings(cross\_encoder, corpus, balanced\_pool, query\_text, batch\_size)

## Resultados y análisis

Se presentan a continuación los resultados obtenidos al evaluar los tres métodos: TF-IDF, Bi-Encoder y Cross-Encoder, todos aplicados sobre el mismo corpus y consultas médicas controladas. Se evaluaron con los valores de K = 1, 3, 5 y 10 usando métricas estándar: Precision@K, Recall@K, MRR, F1@K y nDCG.

1. **Modelo Generalista (BAAI/bge-m3)**

9 Tabla 3.1 Resultados de evaluación para TF-IDF, Bi-Encoder y Cross-Encoder (modelo generalista BAAI/bge-m3)

| **Método** | **K** | **Precision** | **Recall** | **MRR** | **F1** | **nDCG** |
| --- | --- | --- | --- | --- | --- | --- |
| TF-IDF | 1 | 0.8000 | 0.40 | 0.8000 | 0.5334 | 0.8000 |
|  | 3 | 0.6000 | 0.90 | 0.8833 | 0.7200 | 0.8694 |
|  | 5 | 0.4000 | 1.00 | 0.8833 | 0.5714 | 0.9222 |
|  | 10 | 0.2000 | 1.00 | 0.8833 | 0.3333 | 0.9222 |
| Bi-Encoder | 1 | 0.9000 | 0.45 | 0.9000 | 0.6000 | 0.9000 |
|  | 3 | 0.6334 | 0.95 | 0.9500 | 0.7600 | 0.9307 |
|  | 5 | 0.4000 | 1.00 | 0.9500 | 0.5714 | 0.9571 |
|  | 10 | 0.2000 | 1.00 | 0.9500 | 0.3333 | 0.9571 |
| Cross-Encoder | 1 | 1.0000 | 0.50 | 1.0000 | 0.6667 | 1.0000 |
|  | 3 | 0.6000 | 0.90 | 1.0000 | 0.7200 | 0.9226 |
|  | 5 | 0.4000 | 1.00 | 1.0000 | 0.5714 | 0.9754 |
|  | 10 | 0.2000 | 1.00 | 1.0000 | 0.3333 | 0.9754 |

Los resultados evidencian una jerarquía clara entre las tres estrategias de recuperación evaluadas. El Cross-Encoder se posiciona como el método más efectivo, logrando un rendimiento perfecto en métricas como MRR (1.0) y alcanzando la mejor puntuación en nDCG@5 (0.9754). Esto indica su capacidad superior para posicionar correctamente los documentos relevantes en los primeros lugares del ranking.

El Bi-Encoder, aunque computacionalmente más eficiente, también supera al enfoque TF-IDF en métricas clave como MRR y nDCG, validando su capacidad para capturar similitud semántica en contextos médicos. Su rendimiento demuestra que puede ser una etapa de preselección útil antes del reranking con Cross-Encoder.

TF-IDF, si bien limitado por su naturaleza léxica, muestra un recall alto (1.0 en k=5), lo que lo convierte en una herramienta útil para la recuperación inicial de candidatos relevantes. Su inclusión en el pipeline híbrido se justifica por su eficiencia y complementariedad.

En conjunto, el uso de un pool balanceado y la estrategia de reranking permiten maximizar la relevancia de los resultados sin incurrir en el alto coste de evaluar todo el corpus con modelos complejos. Estos hallazgos refuerzan la hipótesis de que combinar estrategias densas y dispersas puede ofrecer beneficios sustanciales en tareas de recuperación médica especializada.

1. **Modelo fine-tuneado (epoch4\_MRR0.9717)**

10 Tabla 3.2 Resultados de evaluación para TF-IDF, Bi-Encoder y Cross-Encoder (modelo fine-tuneado bge\_m3\_epochs/epoch4\_MRR0.9717)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method** | **K** | **Precision** | **Recall** | **MRR** | **F1** | **nDCG** |
| TF-IDF | 1 | 0.8000 | 0.40 | 0.8000 | 0.5334 | 0.8000 |
|  | 3 | 0.6000 | 0.90 | 0.8833 | 0.7200 | 0.8694 |
|  | 5 | 0.4000 | 1.00 | 0.8833 | 0.5714 | 0.9222 |
|  | 10 | 0.2000 | 1.00 | 0.8833 | 0.3333 | 0.9222 |
| Bi-Encoder | 1 | 0.9000 | 0.45 | 0.9000 | 0.6000 | 0.9000 |
|  | 3 | 0.6334 | 0.95 | 0.9333 | 0.7600 | 0.9146 |
|  | 5 | 0.3800 | 0.95 | 0.9333 | 0.5428 | 0.9146 |
|  | 10 | 0.2000 | 1.00 | 0.9333 | 0.3333 | 0.9364 |
| Cross-Encoder | 1 | 1.0000 | 0.50 | 1.0000 | 0.6667 | 1.0000 |
|  | 3 | 0.6000 | 0.90 | 1.0000 | 0.7200 | 0.9226 |
|  | 5 | 0.4000 | 1.00 | 1.0000 | 0.5714 | 0.9754 |
|  | 10 | 0.2000 | 1.00 | 1.0000 | 0.3333 | 0.9754 |

Los resultados muestran un comportamiento inesperado: el modelo fine-tuneado no proporciona mejoras sustanciales sobre el modelo generalista BAAI/bge-m3. El Cross-Encoder mantiene rendimiento perfecto (MRR=1.0, nDCG@5=0.9754) en ambas configuraciones, sugiriendo que ya había alcanzado el techo de rendimiento posible.

El Bi-Encoder fine-tuneado presenta métricas ligeramente inferiores (MRR: 0.9500→0.9333, nDCG@5: 0.9571→0.9146), lo que podría indicar sobreajuste al dataset de entrenamiento o la necesidad de un conjunto de evaluación más amplio para detectar mejoras sutiles.

Estos resultados subrayan la robustez del modelo generalista BAAI/bge-m3 en el dominio médico y sugieren que el fine-tuning específico puede requerir validación en conjuntos de datos más extensos (n=100 consultas) para demostrar su efectividad de forma concluyente.

## Conclusiones generales del experimento:

* + **Hipótesis 1**: Superioridad semántica. Confirmada. Tanto el Bi-Encoder como el Cross-Encoder superan sistemáticamente a TF-IDF en métricas de precisión (MRR, nDCG), con mejoras en MRR de hasta 0.95–1.0 frente a 0.88 para TF-IDF, y en nDCG@5 de hasta 0.97 frente a 0.92. Esto demuestra su capacidad para capturar similitudes más allá de coincidencias léxicas.
  + **Hipótesis 2**: Eficiencia del pool balanceado. Confirmada. La estrategia de combinación TF-IDF + Bi-Encoder en el pool balanceado permite mantener un alto rendimiento en Cross-Encoder sin evaluar el corpus completo, optimizando costes computacionales sin pérdida de calidad.
  + **Hipótesis 3**: Impacto del fine-tuning clínico. Parcialmente confirmada. Aunque el modelo fine-tuneado mostró ligeras mejoras en el Bi-Encoder, los beneficios no fueron significativos respecto al modelo generalista. En el caso del Cross-Encoder, el rendimiento se mantuvo perfecto en ambas versiones, lo que sugiere que el fine-tuning no aporta mejoras sustanciales bajo las condiciones evaluadas (n=10 consultas). La validación completa requiere evaluación sobre el conjunto extendido.

**Conclusión general:**.

Los resultados validan empíricamente el diseño de un sistema RAG basado en Cross-Encoder con pool balanceado como arquitectura óptima para aplicaciones médicas. El pipeline híbrido que integra recuperación léxica (TF-IDF) y densa (Bi-Encoder) seguido de reranking contextual (Cross-Encoder) logra el balance ideal entre precisión y eficiencia computacional.

El análisis comparativo demuestra que el modelo generalista BAAI/bge-m3 proporciona rendimiento robusto y óptimo para el dominio médico evaluado, mientras que el fine-tuning específico no aporta mejoras sustanciales. Esta evidencia refuerza la validez de utilizar el modelo base para la implementación del sistema final, simplificando la arquitectura sin comprometer la calidad de recuperación.

Los hallazgos proporcionan la base empírica sólida para proceder con la implementación del sistema RAG completo del Capítulo 4.

## Experimento 2: BM25 + Bi-Encoder + Cross-Encoder a Nivel de Chunks

Este experimento implementa un pipeline de recuperación a nivel de fragmento (chunk), permitiendo localización precisa de evidencias específicas dentro de documentos extensos, combinando enfoques dispersos y densos..

Este experimento implementa un pipeline de recuperación a nivel de fragmento (chunk), permitiendo la localización precisa de evidencias específicas dentro de documentos extensos.

Se combinan enfoques dispersos (BM25) y densos (Bi-Encoder y Cross-Encoder) sobre un mismo corpus de documentos clínicos.

Se evalúan dos configuraciones: un pipeline secuencial BM25 → Bi-Encoder → Cross-Encoder, y una estrategia paralela basada en pool balanceado entre BM25 y Bi-Encoder.

**Diferencias clave entre TF-IDF y BM25:**

* **TF-IDF (Term Frequency – Inverse Document Frequency)** pondera la importancia de cada término en un documento según:
  + Su frecuencia en el propio documento (TF).
  + Su rareza en la colección (IDF), penalizando términos comunes.
* **BM25** mejora este esquema con dos mecanismos fundamentales:
  + **Saturación de frecuencia:** Limita la ganancia de relevancia que aporta la repetición de una palabra. Es decir, repetir un término muchas veces no incrementa su peso indefinidamente, evitando sobreajuste.
  + **Normalización por longitud de documento:** Penaliza documentos más largos que podrían incluir muchas palabras irrelevantes, equilibrando la puntuación frente a textos más concisos y específicos.

## Objetivos y Hipótesis

**Objetivo principal:** Evaluar si la granularidad a nivel de chunk mejora la precisión de recuperación comparado con documentos completos, y determinar la configuración óptima para localización de evidencias médicas específicas

**Hipótesis de trabajo**:

1. **Superioridad semántica**: Los métodos densos (Bi-Encoder y Cross-Encoder) superan a BM25 en precisión semántica, capturando relaciones profundas de significado.
2. **Eficiencia del pool balanceado**: La estrategia paralela con pool balanceado entre BM25 y Bi-Encoder permite al Cross-Encoder alcanzar alto rendimiento sin evaluar todo el corpus.
3. **Impacto del fine-tuning clínico**: El modelo fine-tuneado mejora significativamente respecto al generalista en MRR, nDCG y precision@K en las fases densas.

**Justificación:**

El uso de chunks permite mayor granularidad y es esencial para sistemas RAG. BM25 ofrece velocidad y cobertura léxica, mientras que Bi-Encoder y Cross-Encoder capturan semántica profunda. Comparar sus combinaciones permite optimizar rendimiento y coste.

## Metodología específica

1. **BM25 Independiente**: Ranking sobre chunks usando representaciones dispersas.

**3.9.1 Objetivos y Hipótesis**

* **Objetivo: Evaluar métodos híbridos sobre chunks**
* **Hipótesis: Cross-Encoder y Bi-Encoder deberían mejorar la precisión local**
* **Justificación: Chunking permite foco más preciso en evidencias**

**3.9.2 Metodología Específica**

* **BM25, Bi-Encoder y Cross-Encoder aplicados sobre fragmentos**
* **Ranking por chunk\_id, chunk\_text, chunk\_position**

**3.9.3 Resultados y Análisis**

**a) Modelo Generalista**

* **Efecto del chunking en recuperación**
* **Limitaciones en precisión semántica**

**b) Modelo Fine-Tuneado**

* **Mejora en recall y relevancia de evidencias**
* **Análisis de casos**

**3.10 Comparativa Final de Experimentos**

* **Tabla resumen de métricas por método y modelo**
* **Análisis cruzado de resultados**
* **Identificación del mejor pipeline para RAG**

Referencias

Alsentzer, E., Murphy, J., Boag, W., Weng, W. H., Jin, D., Naumann, T., & McDermott, M. (2019). [Publicly available clinical BERT embeddings](https://arxiv.org/pdf/1904.03323).

Carrino, C. P., Llop, J., Pàmies, M., Gutiérrez-Fandiño, A., Armengol-Estapé, J., Silveira-Ocampo, J., Valencia, A., Gonzalez-Agirre, A., & Villegas, M. (2022). *Pre-trained Biomedical Language Models for Clinical NLP in Spanish*. Proceedings of the BioNLP 2022 Workshop, 193–199. <https://aclanthology.org/2022.bionlp-1.19.pdf>

Yapi Kredi Teknoloji. (2023, julio 26). *How to improve retrieval performance in RAG*. Medium. <https://medium.com/yapi-kredi-teknoloji/how-to-improve-retrieval-performance-in-rag-e63ea641aa0c>

Kachari, B. (2024). *From search to synthesis: Enhancing RAG with BM25 and Reciprocal Rank Fusion*. Medium. <https://medium.com/@kachari.bikram42/from-search-to-synthesis-enhancing-rag-with-bm25-and-reciprocal-rank-fusion-872d21dc4ca7>

**Anexo A: Formación Complementaria**

**A.1 Cursos Especializados en RAG y LangChain**

Durante el desarrollo de este TFM, se completaron los siguientes cursos especializados de la plataforma DeepLearning.AI para profundizar en las técnicas de recuperación avanzada y procesamiento de lenguaje natural:

**Curso 1: LangChain for LLM Application Development**

* **URL**: https://learn.deeplearning.ai/courses/langchain/lesson/u9olq/introduction
* **Contenidos principales**:
  + Models, Prompts and parsers
  + Memory
  + Chains
  + Question and Answer
  + Evaluation
  + Agents

**Curso 2: Advanced Retrieval for AI with Chroma**

* **URL**: <https://learn.deeplearning.ai/courses/advanced-retrieval-for-ai/lesson/kb5oj/introduction>
* **Contenidos principales**:
  + Overview of embeddings-based retrieval
  + Pitfalls of retrieval- when simple vectors fails
  + Query Expansion
  + Cross-Encoder reranking
  + Embedding adaptors
  + Other Techniques

**Curso 3: LangChain Chat with Your Data**

* **URL**: https://learn.deeplearning.ai/courses/langchain-chat-with-your-data/lesson/snupv/introduction
* **Contenidos principales**:
  + Document Loading
  + Vectorstores and Emebdding
  + Retrieval
  + Question answering
  + Chat