Capítulo 4: Evaluación Experimental del Sistema RAG Médico

**4.1 Metodología de Evaluación**

La evaluación de sistemas RAG requiere un enfoque sistemático que permita comparar objetivamente diferentes estrategias de recuperación y generación. En este trabajo se implementó una metodología experimental rigurosa basada en métricas estándar de recuperación de información y validación estadística.

Todos los scripts, configuraciones y códigos fuente utilizados en los experimentos de este capítulo están disponibles en el repositorio público: <https://github.com/Nagore2021/TFM>

IR ->"Introduction to Information Retrieval"

**4.1.1 Dataset de Evaluación**

El dataset de evaluación (dataset\_test.json) contiene 322 queries (teminología médica y terminologóa coloquial) cubriendo 18 subcategorías. El **Corpus documental** contiene 44 documentos especializados

|  |  |  |
| --- | --- | --- |
| Categoría | Subcategoría | Nº Queries |
| neurologia | ictus | 12 |
| neurologia | epilepsia | 15 |
| enfermedades\_comunes | gripe | 14 |
| cancer | cancer\_colorectal | 26 |
| cancer | cancer\_mama | 27 |
| cancer | cancer\_pulmon | 7 |
| cardiologia | angina\_infarto | 5 |
| cardiologia | hipertension\_arterial | 13 |
| cardiologia | mareo\_sincope\_parada | 2 |
| its-enfermedades transmision sexual | vih\_sida | 28 |
| endocrinologia | diabetes | 25 |
| respiratorias | bronquiolitis | 18 |
| respiratorias | epoc | 21 |
| salud mental | agorafobia | 14 |
| salud mental | esquizofrenia | 26 |
| salud mental | depresion | 27 |
| salud mental | trastorno bipolar | 23 |
| salud mental | ansiedad | 19 |

**4.1.2 Métricas de Evaluación**

Se implementaron las métricas estándar de recuperación de información siguiendo las mejores prácticas establecidas para sistemas RAG (Yapi Kredi Teknoloji, 2024), evaluadas para K ∈ {1, 3, 5, 10}:

**Métricas de Evaluación**

* **nDCG**: ¿Qué tan bien ordenados están los resultados por relevancia?
* **K:** número de documentos a considerar en el ranking para evaluar las métricas.
* **Precision@K**: ¿Qué porcentaje de los K documentos recuperados son relevantes? proporción de documentos relevantes entre los K primeros resultados
* **Recall@K**: ¿Qué porcentaje de documentos relevantes se encontraron en el top-K?
* **F1@K**: media armónica entre precisión y recall.
* **MRR (Mean Reciprocal Rank)**: ¿En qué posición aparece el primer documento relevante?Reciprocal Rank del primer documento relevante (1 si el primero es relevante, 1/2 si el segundo, etc)
* **nDCG@K (Normalized Discounted Cumulative Gain)**: mide relevancia ponderada por posición. Mide la relevancia y orden de los documentos recuperados. : ¿Qué tan bien ordenados están los resultados por relevancia?
* TP / FP / FN: se calculan por query, comparando los documentos recuperados frente al conjunto de documentos relevantes (relevant\_docs) definidos en el dataset:
  + TP (True Positives): documentos recuperados que también están en el ground truth.
  + FP (False Positives): documentos recuperados que no están en el ground truth.
  + FN (False Negatives): documentos relevantes que no fueron recuperado

Estas métricas permiten evaluar simultáneamente precisión, cobertura y orden en la recuperación.

**4.1.3 Validación**

* **Dataset de Evaluación**
* **Traducción automática**: cuando una query en castellano está vacía (text\_es), se traduce automáticamente desde euskera (text\_eu) utilizando el modelo **Helsinki-NLP/opus-mt-eu-es**. Esta traducción se realiza con Hugging Face Pipeline (translation), cargado desde config.yaml:

*text = q.get("text\_es") or translator([q["text\_eu"]])[0]['translation\_text']*

*translator = hf\_pipeline('translation', model=self.cfg['model']['translation\_model'], device=0 if DEVICE=="cuda" else -1)*

Configuración YAML:

*model:*

*translation\_model: "Helsinki-NLP/opus-mt-eu-es"*

* **Reproducibilidad**: todos los experimentos están definidos en scripts parametrizables vía config.yaml
* **Resultados**: Los resultados de cada experimento son almacenados en la carpeta resultados/ para trazabilidad y comparación.
* **Estructura de archivos de resultados**:

Cada experimento genera dos archivos .xlsx diferenciados:

| **Tipo de archivo** | **Descripción** | **Columnas principales** |
| --- | --- | --- |
| \*\_resumen.xlsx | Métricas agregadas por método y valor de K | method, k, precision, recall, f1, mrr, ndcg |
| \*\_detalles.xlsx | Resultados a nivel de query individual | precision, recall, f1, mrr, ndcg, fp, fn, topk, method, query, k, chunk\_position, chunk\_id, chunk\_text |

Descripción de columnas adicionales de Excel detallado para cada experimento:

|  |  |
| --- | --- |
| **Columna** | **DescripcióN** |
| Method | Estrategia de recuperación: biencoder, bm25, hybrid, rerank |
| Query | La pregunta evaluada |
| K | Tamaño del top-k considerado en esa métrica (1, 3, 5, 10) |
| chunk\_position | Posición del chunk dentro del documento (por ejemplo 20/67 = chunk 20 de 67) |
| chunk\_idI | ID completo del chunk recuperado con mayor score |

Esta doble estructura permite realizar análisis tanto cuantitativo como cualitativo de cada método evaluado.

Los experimentos se realizaron tanto con el modelo generalista (BAAI/bge-m3) como con el modelo fine-tuneado (bge\_m3\_epochs/epoch4\_MRR0.9717).

**4.X.1 Objetivos y Hipótesis**

* **Objetivo principal**: declaración clara del propósito específico
* **Hipótesis**: proposición verificable y medible
* **Justificación**: fundamentación técnica y científica

**4.X.2 Metodología Específica**

* Descripción detallada del método/pipeline

**4.X.3 Resultados y Análisis**

**Resultados y Análisis - Modelo Generalista (BAAI/bge-m3)**

* Resultados cuantitativos con métricas clave
* Análisis comparativo con experimentos previos
* Identificación de fortalezas y limitaciones
* Casos de uso donde el método destaca

**Resultados y Análisis - Modelo Fine-tuneado (bge\_m3\_epochs/epoch4\_MRR0.9717)**

**4.2 Experimento 1: TF-IDF + Bi-Encoder + Cross-Encoders**

Este experimento, implementado en el script **1-Tfidf-CrossEncoder\_biencoder.py**, implementa un pipeline híbrido de tres etapas que aprovecha las fortalezas complementarias de diferentes técnicas de recuperación de información.

En este experimento se utilizan documentos completos de tal forma que todos los métodos evalúen exactamente el mismo contenido. Así, cualquier diferencia en resultados se debe únicamente a la **capacidad de la técnica de recuperación**, no a diferencias en qué fragmento del documento se está evaluando.

Para reducir tiempos de inferencia, se aplicaron dos estrategias:

* **Reducir el tamaño del lote (batch\_size) a 4**, limitando el uso de memoria.
* **Reducir el número de documentos a reordenar de 10 a 5** (pool\_size), acelerando el proceso de evaluación sin comprometer significativamente la precisión en el top-k.

**4.2.1 Objetivos y Hipótesis**

* **Objetivo principal**Evaluar y comparar de forma independiente la capacidad de tres enfoques de recuperación de información —TF-IDF, Bi-Encoder y Cross-Encoder— aplicados sobre un mismo corpus médico. El propósito es determinar cuál de ellos ofrece mayor precisión y relevancia al recuperar documentos en respuesta a una consulta médica.
* **Hipótesis**Se espera que los métodos basados en representaciones densas (Bi-Encoder y Cross-Encoder) superen en rendimiento al enfoque basado en representación dispersa (sparse) como TF-IDF, gracias a su capacidad para capturar similitudes semánticas profundas, más allá de las coincidencias léxicas.
* **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.

**4.2.2 Metodología Específica**

Este experimento implementa una comparación independiente entre tres métodos de recuperación sobre un mismo corpus médico, todos evaluados bajo condiciones controladas (mismas consultas, mismo contenido). Se midieron métricas estándar como Precision@k, Recall@k, MRR y nDCG.

Además, el experimento se realiza con **dos variantes de modelos densos**:

* Un modelo **generalista multilingüe** (BAAI/bge-m3).
* Un modelo **fine-tuneado específicamente para textos médicos (bge\_m3\_epochs/epoch4\_MRR0.9717**), entrenado sobre un conjunto de pares pregunta-respuesta derivados del dominio clínico.

1. **Etapa 1: Recuperación Inicial con TF-IDF (sparse)**

* **Descripción:**

TF-IDF transforma cada documento del corpus y cada consulta médica en vectores dispersos (sparse vectors) que reflejan la frecuencia y especificidad de los términos. Luego se calcula la similitud coseno para obtener un ranking de documentos más relevantes.

Aunque no capta relaciones semánticas profundas, TF-IDF es útil para obtener rápidamente un conjunto inicial de documentos potencialmente relevantes.

* **Objetivo en el pipeline:**

Generar un primer ranking de documentos para cada consulta, a partir del cual se extraerá un subconjunto reducido (top-5) que será reordenado por métodos más avanzados (Bi-Encoder y Cross-Encoder).

* **Preparación del corpus:**

Se extraen los textos de cada documento y se genera la matriz TF-IDF sobre todo el corpus:

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

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

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

* **Parámetros utilizados en el vectorizador:**

*vectorizer = TfidfVectorizer(*

*ngram\_range=(1, 2), # Unigramas y bigramas*

*max\_features=15000, # Hasta 15.000 términos (suficiente cobertura)*

*stop\_words=None, # No se eliminan stop words médicas*

*lowercase=True, # Conversión a minúsculas*

*strip\_accents='unicode', # Normalización de acentos*

*min\_df=2, # Palabras que aparecen en al menos 2 documentos*

*max\_df=0.95 # Se excluyen términos muy comunes (>95%)*

*)*

* **Implementación técnica del ranking TF-IDF:**

Para cada consulta, se transforma en vector TF-IDF y se calcula la similitud coseno de cada pregunta con todos los documentos. Se ordena en orden de similitud

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

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

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

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

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

* **Uso en el pipeline:**

*Esta función se usa en el procesamiento de cada consulta. Se selecciona el top-5 de documentos más similares para continuar con las etapas densas.*

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

*pool = tfidf\_ranking[:5] # Reducido de 10 a 5 documentos para acelerar el experimento*

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

* **Descripción:**

En esta fase, se evalúa cada documento mediante un modelo **Bi-Encoder** que transforma tanto la consulta como los documentos en **vectores densos** (dense vectors) en un espacio semántico compartido.

Luego se calcula la **similitud coseno** entre estos vectores para obtener un nuevo ranking de documentos.

Este enfoque es capaz de capturar relaciones de significado más allá de la coincidencia literal de palabras, como sucede en métodos dispersos (sparse) como TF-IDF.

El Bi-Encoder genera las representaciones de la consulta y del documento **por separado** (independientemente).

Esto significa que **no modela directamente cómo se relacionan ambos textos en conjunto**, lo cual es una limitación frente al Cross-Encoder, que sí lo hace. Aun así, es mucho más eficaz que TF-IDF para detectar similitudes semánticas entre términos clínicos que no coinciden literalmente.

* **Modelos utilizados:**

El experimento se realizará con dos versiones del Bi-Encoder para comparar su rendimiento:

* + Un modelo **generalista multilingüe** (BAAI/bge-m3)
  + Una versión **ajustada específicamente al dominio médico** mediante fine-tuning sobre corpus anotado (bge\_m3\_epochs/epoch4\_MRR0.9717).
* **Preparación de datos:**

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

Primero se extraen los IDs y textos del corpus:

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

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

* **Implementación técnica del cálculo Bi-Encoder:**El Bi-Encoder transforma la consulta y todos los documentos en vectores densos, luego calcula la similitud coseno para obtener el ranking final:

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

*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]*

*doc\_embs = biencoder.encode(docs\_texts, convert\_to\_tensor=True, device=device)*

*similarities = []*

*if hasattr(q\_emb, 'cpu'): # GPU*

*for doc\_emb in doc\_embs:*

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

*similarities.append(sim)*

*else: # CPU*

*for doc\_emb in doc\_embs:*

*sim = float(np.dot(q\_emb, doc\_emb) / (np.linalg.norm(q\_emb) \* np.linalg.norm(doc\_emb)))*

*similarities.append(sim)*

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

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

Primero, la consulta se transforma en un embedding denso:

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

Luego, se extraen los textos del pool y se codifican:

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

*doc\_embs = biencoder.encode(docs\_texts, convert\_to\_tensor=True, device=device)*

La similitud coseno se calcula entre el vector de la consulta y cada embedding de documento:

*for doc\_emb in doc\_embs:*

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

*similarities.append(sim)*

Finalmente, los documentos se ordenan de mayor a menor similitud:

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

*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)*

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

* **Descripción:**

En la etapa final 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 entre ambos textos.

Esta representación densa está **contextualizada**, porque el modelo considera el modelo considera no solo qué palabras aparecen, sino **cómo interactúan en el contexto conjunto**

* **Objetivo:**

Aplicar una evaluación más precisa de la relevancia semántica mediante un modelo Cross-Encoder.

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

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.

* **Modelo utilizado:**

Se emplea el modelo **BAAI/bge-reranker-v2-m3**, un **Cross-Encoder generalista multilingüe** diseñado específicamente para tareas de reranking. Este modelo es complementario al Bi-Encoder BAAI/bge-m3, lo que permite evaluar de forma coherente tanto la codificación independiente (Bi-Encoder) como la conjunta (Cross-Encoder).

* **Implementación técnica:**
  + **Construcción del pool balanceado:**

Para garantizar que el Cross-Encoder trabaje sobre un conjunto diverso de candidatos y no herede sesgos de una única estrategia previa, se construye un "pool balanceado" de documentos:

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

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):*

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

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

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

Se van alternando documentos de ambas listas, evitando duplicados (seen), hasta alcanzar el número total deseado (pool\_size), lo que reduce la posibilidad de sesgo metodológico:

*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*

*return balanced\_pool*

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)*

El Cross-Encoder (BAAI/bge-reranker-v2-m3) en esta función **calculate\_crossencoder\_rankings ()** evalúa pares consulta-documento directamente, capturando relaciones semánticas profundas en contexto cruzado. Finalmente, los documentos se ordenan por puntuación de relevancia

*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) # El Cross-Encoder devuelve una* ***lista de scores****, uno por cada par de (preguntas, respuesta). Se hacen menos llamadas al Cross Encoder*

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

1. **Etapa 4: Pipeline Híbrido (TF-IDF → Bi-Encoder → Cross-Encoder)**

* **Descripción general:**

Esta estrategia combina las tres metodologías anteriores en cascada, aprovechando lo mejor de cada una para lograr un sistema de recuperación más eficaz.

Primero se usa TF-IDF para filtrar rápidamente un subconjunto inicial del corpus (eficiencia léxica), luego un Bi-Encoder reordena ese subconjunto (precisión semántica) y finalmente un Cross-Encoder realiza una evaluación exhaustiva de relevancia (precisión contextualizada).

* **Implementación técnica**
  1. **Filtro inicial con TF-IDF**

Primero se seleccionan los documentos más relevantes según su similitud léxica con la consulta (representación dispersa):

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

* 1. **Reordenación semántica con Bi-Encoder**

Luego se calcula el embedding denso de la consulta y de los documentos del tfidf\_pool:

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

pool\_texts = [corpus[doc\_id] for doc\_id in tfidf\_pool]

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

Se calcula la similitud coseno entre el embedding de la consulta y cada documento:

*sem\_scores = [*

*float(torch.cosine\_similarity(q\_emb.unsqueeze(0), emb.unsqueeze(0)))*

*for emb in pool\_embs*

Y se reordena el tfidf\_pool por similitud semántica:

*biencoder\_reranked = [doc for \_, doc in sorted(zip(sem\_scores, tfidf\_pool), key=lambda x: x[0], reverse=True)]*

* 1. **Reranking final con Cross-Encoder**

Por último, el Cross-Encoder evalúa directamente cada par (consulta, documento):

*final\_ranking = calculate\_crossencoder\_rankings(*

*cross\_encoder, corpus, biencoder\_reranked, query\_text, batch\_size*

*)*

**4.2.3 Resultados y Análisis**

Para cada método (TF-IDF, Bi-Encoder, Cross-Encoder y Pipeline Híbrido), se evaluaron los documentos recuperados frente al *ground truth* utilizando las métricas estándar de recuperación:

* Precision@k: Porcentaje de documentos relevantes en el top-k.
* Recall@k: Porcentaje de documentos relevantes recuperados respecto al total.
* F1@k: Media armónica entre precisión y recall.
* MRR (Mean Reciprocal Rank): Inverso de la posición del primer resultado relevante.
* nDCG@k: Relevancia acumulada, ponderada por la posición.

Se analizaron los resultados de los cuatro métodos para distintos valores de **k** (1, 3, 5, 10).

Aunque el ground truth contiene en su mayoría una única respuesta relevante por consulta, se ha optado por mantener K=5 en las métricas para reflejar tanto la exhaustividad como la calidad del ranking, y asegurar comparabilidad con futuros escenarios o ground truths ampliados.

**4.2.3.1** Análisis Comparativo **Modelo Generalista (BAAI/bge-m3)**

Las siguientes observaciones destacan las tendencias más relevantes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Análisis Comparativo Detallado- Modelo Generalista (BAAI/bge-m3)** | | | | | | |
| **method** | **k** | **precision** | **recall** | **mrr** | **f1** | **ndcg** |
| tfidf | 1 | 0,8 | 0,4 | 0,8 | 0,5334 | 0,8 |
| tfidf | 3 | 0,6 | 0,9 | 0,8833 | 0,72 | 0,8694 |
| tfidf | 5 | 0,4 | 1 | 0,8833 | 0,5714 | 0,9222 |
| tfidf | 10 | 0,2 | 1 | 0,8833 | 0,3333 | 0,9222 |
| biencoder | 1 | 0,9 | 0,45 | 0,9 | 0,6 | 0,9 |
| biencoder | 3 | 0,6334 | 0,95 | 0,95 | 0,76 | 0,9307 |
| biencoder | 5 | 0,4 | 1 | 0,95 | 0,5714 | 0,9571 |
| biencoder | 10 | 0,2 | 1 | 0,95 | 0,3333 | 0,9571 |
| cross\_encoder | 1 | 1 | 0,5 | 1 | 0,6667 | 1 |
| cross\_encoder | 3 | 0,6 | 0,9 | 1 | 0,72 | 0,9226 |
| cross\_encoder | 5 | 0,4 | 1 | 1 | 0,5714 | 0,9754 |
| cross\_encoder | 10 | 0,2 | 1 | 1 | 0,3333 | 0,9754 |
| hybrid\_pipeline | 1 | 1 | 0,5 | 1 | 0,6667 | 1 |
| hybrid\_pipeline | 3 | 0,6 | 0,9 | 1 | 0,72 | 0,9226 |
| hybrid\_pipeline | 5 | 0,4 | 1 | 1 | 0,5714 | 0,9754 |
| hybrid\_pipeline | 10 | 0,2 | 1 | 1 | 0,3333 | 0,9754 |

*Tabla 1: Análisis Comparativo Detallado*

Los resultados muestran una jerarquía clara y coherente entre los métodos evaluados. El **Cross-Encoder y el Pipeline Híbrido logran un rendimiento idéntico**, situándose como las estrategias más precisas. Sin embargo, el Pipeline Híbrido introduce mayor complejidad sin ofrecer beneficios adicionales. Por tanto, el **Cross-Encoder con pool balanceado** se posiciona como el enfoque más efectivo.

La siguiente tabla resume el rendimiento de los tres enfoques evaluados (TF-IDF, Bi-Encoder y Cross-Encoder) utilizando métricas estándar de recuperación. Se destaca la mejora relativa del Cross-Encoder respecto al baseline TF-IDF:

| **Métrica** | **TF-IDF** | **Bi-Encoder** | **Cross-Encoder** | **Mejora vs TF-IDF** | **% Mejora** |
| --- | --- | --- | --- | --- | --- |
| **Precision@5** | 0.40 | 0.40 | 0.40 | 0.00 | 0.0% |
| **Recall@5** | 1.00 | 1.00 | 1.00 | 0.00 | 0.0% |
| **F1@5** | 0.5714 | 0.5714 | 0.5714 | 0.00 | 0.0% |
| **MRR@5** | 0.8833 | 0.95 | **1.0000** | +0.1167 | **+13.2%** |
| **nDCG@5** | 0.9222 | 0.9571 | **0.9754** | +0.0532 | **+5.8%** |

*Tabla 2: Comparativa de rendimiento (K = 5)*

Las tres estrategias alcanzan el mismo nivel en Precision@5, Recall@5 y F1@5. Esto indica que, aunque todos los métodos recuperan el documento relevante dentro del top-5, **ninguno mejora el número total de documentos relevantes encontrados ni la proporción en ese top-k**.

**MRR (Mean Reciprocal Rank)**: Aquí el Cross-Encoder logra un rendimiento perfecto (1.0), lo que significa que **en todas las consultas, el primer documento relevante se encuentra en la primera posición** del ranking. Esto representa una mejora significativa del **+13.2% respecto a TF-IDF**, que tiende a colocar los documentos relevantes en posiciones inferiores.

**nDCG@5**: El Cross-Encoder también supera a los demás métodos en esta métrica, con una mejora del **+5.8%** frente a TF-IDF. Esto refleja una **mejor calidad del ordenamiento**, premiando más a los documentos relevantes que aparecen antes en la lista.

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

**4.3.1 Objetivos y Hipótesis**

**Objetivo principal:**Evaluar si un pipeline progresivo que combina BM25, Bi-Encoder y Cross-Encoder a nivel de chunks mejora la precisión de recuperación en contexto RAG médico, identificando el fragmento más relevante del documento

El objetivo específico es comparar dos variantes del Bi-Encoder: una versión generalista multilingüe (BAAI/bge-m3) y una versión especializada tras fine-tuning (bge\_m3\_epochs/epoch4\_MRR0.9717) entrenada sobre corpus médicos.

**Hipótesis:**

Se espera que esta combinación permita localizar con mayor precisión los \*chunks\* más relevantes que responden a una consulta médica, superando los enfoques tradicionales basados en documentos completos o \*chunk0\* por defecto.

**Justificación:**Trabajar a nivel de chunk mejora la granularidad de la recuperación. BM25 actúa como primer filtro eficiente léxico, el Bi-Encoder permite una reordenación semántica basada en similitud coseno, y el Cross-Encoder refina la clasificación con precisión token a token.

Esta arquitectura progresiva ha sido sugerida en trabajos recientes para sistemas RAG más precisos y eficientes (Kachari, 2024).

**4.2.2 Metodología Específica**

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

* TF-IDF pondera los términos según su frecuencia en el documento y su rareza en la colección (frecuencia inversa de documento).
* BM25 mejora este esquema al aplicar:
  + **Saturación de frecuencia:** La ganancia por repetición de una palabra se estabiliza en lugar de crecer linealmente.
  + **Normalización por longitud:** Penaliza documentos excesivamente largos que podrían contener muchas palabras irrelevantes.

**Descripción detallada del método/pipeline**

1. BM25 a nivel de **chunk**: Recupera los 10 mejores chunks (no documentos) basándose en frecuencia de términos y normalización.

*bm25\_pool = self.search\_bm25\_initial(q, pool\_size=10)*

1. Bi-Encoder: Genera embeddings del query y del contenido del chunk y reordena según similitud coseno.

*biencoder\_reranked = self.search\_biencoder\_rerank(q, bm25\_pool)*

1. Cross-Encoder: Evalúa pares query-chunk usando atención cruzada para obtener un reranking final de alta precisión.

*final\_ranking = self.search\_crossencoder\_final(q, biencoder\_reranked)*

**Preprocesamiento de consultas**:

* + Traducción automática del euskera al español si es necesario.
  + Normalización de texto con spaCy (eliminación de signos, espacios, minúsculas).

*def load\_queries(json\_path, translator):*

*with open(json\_path, 'r', encoding='utf-8') as f:*

*data = json.load(f)*

*queries, relevant = [], {}*

*for q in data['queries']:*

*text = q.get("text\_es") or translator([q["text\_eu"]])[0]['translation\_text']*

*if text:*

*norm = normalizar\_doc(text)*

*queries.append(norm)*

*relevant[norm] = [normalize\_doc\_id(d) for d in q['relevant\_docs']]*

*return queries, relevant*

**Indexación en ChromaDB:**

- Cada documento está segmentado en \*chunks\* de 300 caracteres con metadatos (`chunk\_id`, `chunk\_position`, `document\_id`).

- BM25 trabaja directamente sobre los \*chunks\*, no sobre documentos agregados.

*data = self.collection.get(include=['documents', 'metadatas'])*

*self.docs\_raw = dict(zip([m['chunk\_id'] for m in data['metadatas'] if 'chunk\_id' in m], data['documents']))*

*self.docs\_norm = [normalizar\_doc(doc) for doc in data['documents']]*

**Selección del mejor chunk:**

Durante la búsqueda inicial con BM25 se agrupan los chunks por documento y se selecciona aquel con mayor score:

*# Agrupar chunks por documento y encontrar el de mayor score BM25*

*temp = defaultdict(list)*

*for i, score in enumerate(scores):*

*chunk\_id = self.chunk\_ids[i]*

*doc\_id = normalize\_doc\_id(self.metadatas[chunk\_id]['document\_id'])*

*temp[doc\_id].append((score, chunk\_id))*

*# Seleccionar mejor chunk para cada documento*

*top\_score, best\_chunk = max(scores\_chunks, key=lambda x: x[0])*

*self.doc\_to\_best\_chunk[doc\_id] = best\_chunk*

Este chunk seleccionado se utiliza en las etapas posteriores del pipeline (Bi-Encoder y Cross-Encoder), asegurando coherencia y máxima relevancia.

\*\*Etapa 2 - Bi-Encoder:\*\*

Se utiliza el modelo `BAAI/bge-m3` (o fine-tuneado) para reordenar los documentos seleccionados por BM25. Para ello, se generan embeddings del query y del contenido del mejor chunk de cada documento:

```python

# Embeddings y cálculo de similitud coseno

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

pool\_embs = self.biencoder.encode(pool\_texts, convert\_to\_tensor=True)

similarities = [float(torch.cosine\_similarity(q\_emb.unsqueeze(0), emb.unsqueeze(0))) for emb in pool\_embs]

```

Este ranking es más semántico y suele subir el MRR respecto a BM25.

\*\*Etapa 3 - Cross-Encoder:\*\*

Se genera una predicción más precisa comparando el texto del chunk y el query token a token. Se usa `BAAI/bge-reranker-v2-m3`:

```python

# Reranking final con Cross-Encoder

pairs = [[query, chunk\_text] for chunk\_text in pool\_texts]

scores = self.cross\_encoder.predict(pairs)

```

Este método es más costoso computacionalmente pero mucho más preciso, y finaliza el pipeline generando el ranking más afinado.

**Evaluación del rendimiento:**

Para cada combinación de query, método (bm25, biencoder, cross\_encoder) y valor de *k* ([1, 3, 5, 10]), se calculan métricas clave que permiten medir la calidad de recuperación:

*metrics = evaluate\_with\_metadata(*

*retrieved=ranking, # Lista ordenada de documentos recuperados*

*gold=gold, # Documentos relevantes según el ground truth*

*k=k, # Número de resultados evaluados*

*metadata=self.metadatas, # Metadatos de los chunks para análisis contextual*

*method=method\_name, # Nombre del método evaluado*

*query\_text=q # Query original*

*)*

*self.records.append(metrics)*

Esta función evaluate\_with\_metadata() no solo calcula métricas cuantitativas como precision, recall, F1, MRR y nDCG, sino que también almacena información contextual como:

El chunk\_id con mayor puntuación recuperado.

La posición del chunk en el documento (chunk\_position, útil para visualizar en qué parte del documento se encuentra la respuesta).

El texto plano del chunk (chunk\_text) para análisis cualitativo.

**4.2.3 Resultados y Análisis**

##### Modelo Generalista: BAAI/bge-m3

\*\*BM25:\*\*

- Mejora respecto al uso por documento al evaluar directamente \*chunks\*.

- Resultados aún limitados semánticamente (`Precision@1` moderada).

\*\*Bi-Encoder:\*\*

- Reordenación semántica eficaz: sube MRR y Recall significativamente.

- Más robusto en términos clínicos no explícitos.

\*\*Cross-Encoder:\*\*

- Máxima precisión: top-1 suele incluir chunks directamente relacionados con la consulta.

- `nDCG@10 = 1` en múltiples queries, lo que indica buen orden de recuperación.

##### Modelo Fine-tuneado: bge\_m3\_epochs/epoch4\_MRR0.9717

\*\*Bi-Encoder:\*\*

- Resultados superiores en queries clínicas complejas.

- Embeddings más adecuados para el dominio médico.

\*\*Cross-Encoder:\*\*

- Alta precisión incluso en top-1 (`Precision@1 > 0.9` en múltiples queries).

- Excelente desempeño en recall y nDCG.

\*\*Conclusión:\*\*

- Evaluar a nivel de \*chunk\* mejora considerablemente la recuperación.

- La arquitectura progresiva permite combinar velocidad inicial (BM25), semántica (Bi-Encoder) y precisión (Cross-Encoder).

- Es la mejor estrategia identificada hasta ahora para contexto RAG en dominios clínicos.