UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Comparativa de tecnologías de búsqueda en bancos de prueba para el estudio de la desinformación en el ámbito de consultas de la salud

Autor/a:
Xiana Carrera Alonso

Titores:
David E. Losada Carril
Marcos Fernández Pichel

Grao en Enxeñaría Informática

Julio 2025

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Grao en Enxeñaría Informática

D. David Enrique Losada Carril, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **D. Marcos Fernández Pichel**, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela,

INFORMAN:

Que a presente memoria, titulada *Comparativa de tecnologías de búsqueda en bancos de prueba para el estudio de la desinformación en el ámbito de consultas de la salud*, presentada por **Dna. Xiana Carrera Alonso** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo nosa titoría no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 3 de xullo de 2025:

Titor/a, Cotitor/a, Alumno/a,

David Enrique Losada Carril Marcos Fernández Pichel Xiana Carrera Alonso

Agradecimientos

Este trabajo está dedicado a mi madre. Gracias por apoyarme y cuidarme incondicionalmente durante los 23 años que le precedieron. Sin ti, este trabajo no hubiera sido posible. Siento que no pudieras llegar a verlo acabado, pero creo que te hubiera gustado. Te echo mucho de menos.

Quiero dar las gracias a toda mi familia, por su cariño y confianza. En especial, se las doy a mi abuela por su infinita bondad, atención y paciencia conmigo, y por darle sentido al mundo.

Para Óscar no hay palabras remotamente suficientes para expresar mi agradecimiento por caminar a mi lado, por ser como es y por toda su ternura.

Gracias también a los amigos que han compartido conmigo este viaje, estén donde estén, por lo que hemos vivido y por ser mi eterna inspiración.

Este trabajo tampoco hubiera sido posible sin David y Marcos, cuya guía y dedicación a lo largo de este año han sido inestimables, y de los que he aprendido mucho más de lo que cabe en este trabajo.

Resumen

La desinformación presente en los resultados de búsquedas web sobre salud es una cuestión preocupante tanto a nivel social como científico, pues puede influir negativamente en la toma de decisiones de los/las usuarios/as y acarrear graves consecuencias para la salud. Este fenómeno, que cobró especial visibilidad con la pandemia de la COVID-19, es una de las áreas de investigación dentro del ámbito de la Recuperación de Información (RI), en torno a la cual se articula este trabajo.

El proyecto tiene como objetivo central explorar cómo discernir documentos relevantes y correctos de documentos dañinos, dada una cierta intención de búsqueda. Para ello, se propone un estudio estructurado en tres líneas de investigación. En primer lugar, se lleva a cabo un análisis sistemático del desempeño de sistemas de búsqueda del estado del arte con respecto a esta tarea. En segundo lugar, se diseña, implementa y evalúa una nueva técnica basada en Modelos Grandes de Lenguaje (LLMs) para la generación de alternativas de consultas enviadas por usuarios/as, de forma que sus variantes favorezcan la recuperación de documentos relevantes y correctos, en detrimento de la recuperación de documentos dañinos. Por último, se presenta un estudio sobre la predicción de la presencia de desinformación sanitaria en los resultados de búsqueda de consultas, para lo cual se prueban técnicas procedentes de ámbitos relacionados y se diseña, implementa y evalúa un nuevo predictor basado en LLMs y específico para esta tarea.

Los hallazgos del trabajo avalan el potencial de los LLMs en el campo de la RI, al lograr mejorar la eficacia de sistemas de búsqueda punteros. Además, se subsana la ausencia de literatura previa en cuanto a la predicción de desinformación en consultas, a la vez que se prueba la capacidad de los LLMs para ello, superior a la capacidad que demuestran técnicas más generales.

Parte de las aportaciones de este proyecto han sido aceptadas para su publicación en la conferencia ACM SIGIR 2025.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La World Wide Web ha transformado y democratizado el acceso a todo tipo de información. La facilidad y rapidez de acceso a una amplia variedad de fuentes ha supuesto una indudable revolución social y científica, pero estos avances tecnológicos no están exentos de riesgos. La proliferación de desinformación, ya sea intencional o involuntaria, provoca que los resultados de búsquedas digitales puedan contener información incorrecta [1], ocasionando una manifiesta preocupación de los/las usuarios/as por la fiabilidad y veracidad del contenido consumido online [2].

La presencia de desinformación es especialmente relevante en el contexto médico y sanitario, donde se ha comprobado que resultados con información incorrecta pueden provocar que los/las usuarios/as tomen decisiones erróneas [3], con consecuencias potencialmente dañinas para su salud cuando no hay una supervisión médica adecuada [4].

La pandemia de COVID-19 declarada en 2020 dio mayor visibilidad y concienciación al respecto de esta problemática. Se identificaron numerosos rumores, estigmas y teorías conspirativas asociadas a la COVID-19 [5], y organizaciones como la OMS recomendaron una monitorización sistemática y la implantación de medidas de control para confrontarlas [6].

Por otro lado, el hecho de que los motores de búsqueda sean habitualmente utilizados en búsquedas online sobre salud [7] hace de gran interés el estudio de su efectividad a la hora de distinguir resultados correctos y creíbles de información total o parcialmente incorrecta. Esta rama de investigación se enmarca dentro del área de la Recuperación de Información (RI), que se puede definir como el campo dedicado a la "recuperación de material (normalmente documentos) de naturaleza no estructurada (habitualmente textual) que cumple una necesidad

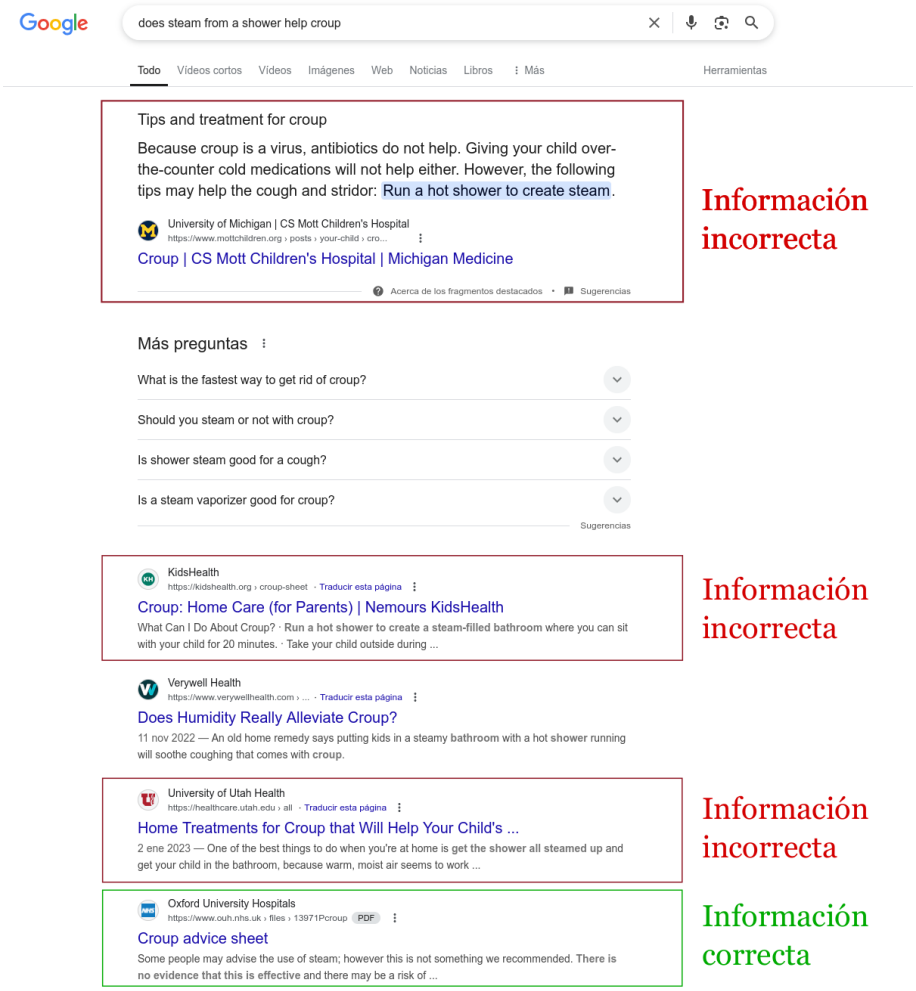


Figura 1.1: Resultados de una búsqueda de Google realizada a partir de una consulta extraída de la TREC 2021 Health Misinformation Track. La búsqueda se realizó en modo incógnito el día 7 de mayo de 2025.

de información dentro de grandes colecciones (generalmente almacenadas en ordenadores)" [8].

Los últimos avances en el área del Procesamiento del Lenguaje Natural (PLN, por sus siglas en inglés) resultan especialmente prometedores para este propósito. El desarrollo de Modelos Grandes de Lenguaje (LLMs) como GPT [9] o BERT [10] ha permitido grandes avances en tareas de PLN como la generación de texto [11] y la comprensión de lenguaje [12], y su uso de forma conversacional ofrece una alternativa a modelos de búsqueda clásicos basados en algoritmos como PageRank. En ocasiones, los sistemas de búsqueda presentan una baja eficacia a la hora de promover sitios web fiables por encima de otros no fiables [13] (con un ejemplo en la Figura 1.1) y, por tanto, es natural plantear el uso de LLMs para

mejorar las capacidades actuales de la tecnología de RI. No obstante, a pesar de los grandes avances realizados (que han supuesto un cambio de paradigma en los campos del PLN y de la RI), se ha comprobado que, en ocasiones, los LLMs producen resultados incorrectos, los cuales pueden estar presentes incluso en salidas coherentes y gramaticalmente correctas, en las comúnmente llamadas "alucinaciones" [14]. Por consiguiente, su aplicación a la hora de proporcionar información relativa al ámbito de la salud está supeditada a un análisis que evalúe, valore y contraponga su eficacia y riesgos.

1.1. Objetivos e Hipótesis

El objetivo principal de este Trabajo de Fin de Grado es llevar a cabo un estudio de diferentes técnicas que pueden ayudar a detectar la desinformación online relacionada con consultas de salud y minimizar la presencia de documentos dañinos en las páginas de resultados. En particular, distinguimos los siguientes objetivos específicos:

(1) Revisar la literatura y últimos avances en el campo de la RI del ámbito de búsquedas relacionadas con la salud.

(2) Evaluar la capacidad de modelos de búsqueda del estado del arte para distinguir entre información *harmful* (relevante pero dañina) e información *helpful* (relevante y no dañina), con el objetivo de recuperar esta última en detrimento de la primera, utilizando conjuntos de datos específicamente diseñados para esta tarea.

(3) Diseñar e implementar una nueva técnica de generación de consultas alternativas, fundamentada en el uso de LLMs, que mejore las puntuaciones de referencia establecidas en el objetivo específico (2). Las consultas originales de los/las usuarios/as pueden describir pobremente las necesidades de información y este objetivo persigue explorar métodos automáticos que puedan reformularlas para recuperar más documentos *helpful* y menos *harmful*.

(4) Evaluar la capacidad de diferentes predictores clásicos para anticipar la calidad de los resultados obtenidos por una determinada consulta. Exploraremos estrategias *pre-retrieval* que, sin realizar ningún proceso de recuperación, basen su predicción en distintas características de las consultas originales. El objetivo será predecir en qué medida una determinada consulta es propensa a recuperar documentos dañinos.

(5) Diseñar e implementar nuevos predictores de la calidad de los resultados de consultas, basados en el uso de LLMs, que mejoren las puntuaciones de referencia establecidas en el objetivo específico (4).

(6) Analizar y extraer conclusiones de los resultados obtenidos, tanto cuantitativa como cualitativamente, cuando lo segundo es posible.

Como parte de la consecución de estos objetivos, se busca también validar las siguientes hipótesis:

(I) Los modelos neuronales de búsqueda del estado del arte que son más eficaces en la RI son también los de más capacidad para discernir entre información *harmful* e información *helpful*.

(II) Los LLMs pueden emplearse para generar consultas alternativas que mejoren el desempeño de modelos de búsqueda del estado del arte en cuanto a la distinción entre información *harmful* e información *helpful*.

(III) La incorporación de narrativas (párrafos descriptivos de las necesidades de información) en la generación de consultas con LLMs ayuda a clarificarlas. En ausencia de narrativas reales, estas pueden ser satisfactoriamente reemplazadas por narrativas sintéticas generadas por LLMs.

(IV) Debido a la relación entre las medidas de *harmfulness* y la relevancia de documentos, los predictores de relevancia de consultas clásicos son capaces de estimar, de manera indirecta, la *harmfulness* de los resultados de búsqueda.

(V) Los LLMs pueden ser empleados para predecir la calidad de resultados de consultas en cuanto a desinformación (específicamente, a través de la evaluación del nivel de controversia de una consulta) y constituyen herramientas prometedoras para mejorar los resultados de predictores clásicos.

Los resultados obtenidos en la consecución del objetivo específico (3), junto a algunos de los análisis pertinentes a los objetivos específicos (2) y (6), fueron aceptados para su publicación como un artículo corto en ACM SIGIR, una conferencia de clase GGS 1, puntuación GGS A++ y ranking CORE A*:

> X. Carrera, M. Fernández-Pichel y D.E. Losada. “Generating Effective Health-Related Queries for Promoting Reliable Search Results”, en: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.

Capítulo 2

Estado del Conocimiento

2.1. Recuperación de Información

El objetivo principal de la RI es "proporcionar la información más relevante al/a la usuario/a en su consulta" [15]. Dado que múltiples documentos pueden ser relevantes, suelen ordenarse en función de una puntuación de similitud con la consulta del/de la usuario/a. Los sistemas de recuperación de texto tradicionales se basan en la coincidencia de términos entre la consulta y el contenido de documentos, pero este enfoque léxico presenta diversas limitaciones, como la polisemia, la sinonimia y el uso de vocabulario diferente para hacer referencia a un mismo concepto en la consulta y en los documentos, fenómeno conocido como *lexical gap* [16].

En los últimos años, los avances en algoritmos de aprendizaje profundo como las *convolutional neural networks* (CNNs) y *recurrent neural networks* (RNNs), junto con técnicas de aprendizaje por transferencia, mecanismos de atención como la arquitectura de transformadores y el uso de modelos de lenguaje preentrenados, como GPT [9] y BERT [10], han contribuido significativamente a mejorar el rendimiento de los sistemas de RI [17]. Otros desarrollos recientes se enfocan en la incorporación de conocimiento externo mediante *knowledge graph embeddings* [18], por ejemplo, así como en la integración de información multimodal que combina texto, imágenes y audio [19].

Lo habitual es aplicar estas técnicas en dos etapas: **recuperación** y **reranking**, representadas en la Figura 2.1. En la primera etapa, se obtiene una colección de documentos potencialmente relevantes para la consulta del/de la usuario/a, ordenados según su puntuación de similitud con ella. Para ello, se emplean técnicas como el modelo Booleano [8], el modelo de espacio vectorial [20],

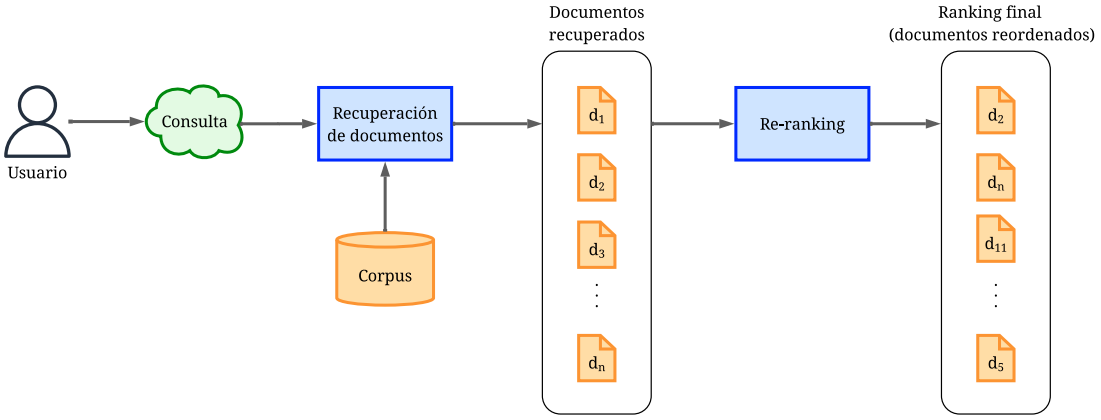
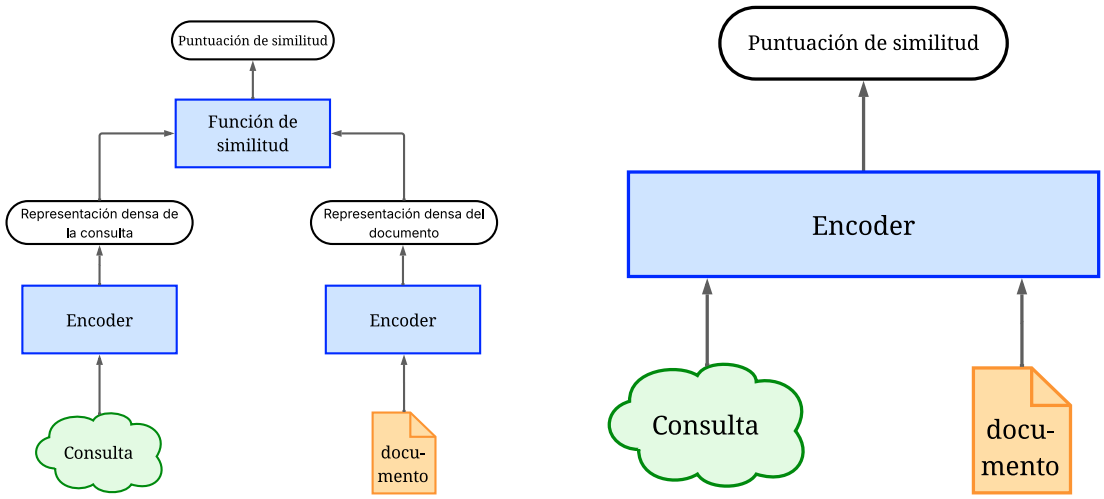


Figura 2.1: Etapas habituales en los sistemas de RI modernos.

BM25 [21] (que supuso una revolución en esta tarea, al tener en cuenta la frecuencia de los términos y variaciones en la longitud de los documentos), Latent Semantic Indexing [22], Latent Dirichlet Allocation [23], o modelos de lenguaje preentrenados como BERT. En la segunda etapa, el ranking inicial se reordena, con el objetivo de refinarlo. Los modelos empleados suelen ser distintos a los de la fase de recuperación, ya que aquí se valora más la eficacia y precisión que la eficiencia computacional. Algunos ejemplos de métodos de re-ranking son RankNet [24], DRMM [25] y Duet [26].

A su vez, las técnicas de recuperación del estado del arte pueden clasificarse en tres categorías principales: dispersas, densas e híbridas. Los métodos **dispersos** representan documentos y consultas mediante vectores de alta dimensionalidad y en los que la mayoría de las componentes son nulas (de modo que se dice que son dispersos). Estos vectores son después comparados con medidas como la similitud coseno o el producto escalar [27]. En una de las formulaciones más básicas, conocida como *bag-of-words*, cada dimensión se corresponde con un término del vocabulario, y su valor indica el número de ocurrencias en el texto del término que representa. Modelos más sofisticados asignan pesos a los términos para capturar su importancia y su significado semántico. Esta ponderación se puede realizar con sistemas neuronales como, por ejemplo, DeepCT [28], uniCOIL [29], Doc2query [30] o la versión mejorada de este último, DocT5query [31], que genera consultas sintéticas para los documentos y los expande con ellas, en un proceso de *data augmentation*. Otro posible enfoque consiste en crear vectores dispersos directamente a través de redes neuronales, abstrayendo la información semántica como elementos de un espacio latente que, aunque sin correspondencia biunívoca con términos del vocabulario, pueden evitar problemas como la sinonimia y la polisemia y pueden ser almacenados y recorridos mediante índices invertidos, al igual que al representar términos de forma convencional. Algunos ejemplos de este enfoque son SNRM [32], UHD-BERT [33] y BPR [34].



(a) *Dual-encoder*. (b) *Cross-encoder*.

Figura 2.2: Comparativa entre las arquitecturas *dual-encoder* y *cross-encoder*.

Algunos modelos dispersos, como el ya mencionado DocT5query, emplean técnicas que enriquecen la información disponible para hacer búsquedas, como extender las consultas o documentos con sinónimos o palabras relevantes para reducir el *lexical gap* (*query/document augmentation*), introducir dependencias entre términos para capturar su orden y las relaciones entre ellos, deducir temáticas de los textos y emparejar consultas y documentos en base a temáticas comunes (*topic model*), entre otras.

Los métodos **densos** están basados, generalmente, en una arquitectura de *dual-encoder*, también conocida como red siamesa y que aparece representada en la Figura 2.2a. Esta consiste de dos redes idénticas que reciben consultas y documentos, respectivamente, y desarrollan representaciones densas para cada uno de forma independiente. Dichas representaciones pertenecen a un espacio latente $\mathbb{R}^d$ con d pequeño, es decir, de pocas dimensiones (por ejemplo, en el caso de BERT se tiene que $d = 768$). Normalmente, estas representaciones son obtenidas mediante una agrupación de las representaciones individuales de los términos que aparecen en el texto, teniendo en cuenta su contexto. Los vectores de $\mathbb{R}^d$ pueden ser comparados mediante su similitud coseno o su producto escalar, por ejemplo. Sobre esta premisa, empleada por modelos como DPR [35], se pueden añadir otras estrategias. Por ejemplo, ANCE [36] refuerza los casos negativos usados durante el entrenamiento de modelos densos, sustituyéndolos por otros de mayor calidad encontrados a través de un índice de vecinos más cercanos. Por su parte, ColBERT [37], un transformador diseñado para tareas de *Open-domain question answering* (OpenQA) da más importancia a la similitud entre

pasajes que a la similitud entre palabras y representa cada token de la entrada ya como un vector denso, posponiendo la interacción entre tokens de la consulta y del documento a etapas avanzadas. Otros estudios plantean el uso de redes neuronales gráficas (*graph neural networks*, GNNs) para la RI, al ser adecuadas para modelar relaciones entre términos, documentos y otras entidades [38, 39].

Los métodos **híbridos** combinan diferentes representaciones, arquitecturas y técnicas. Por ejemplo, Vulić y Moens [40] proponen hacer uso de combinaciones lineales monolingües y multilingües, lo cual da mejores resultados en tareas de recuperación de información en varias lenguas.

En cuanto a la etapa de re-ranking, se distinguen dos categorías principales: modelos de *learning to rank* y neuronales. *Learning to rank* hace referencia a algoritmos supervisados de aprendizaje automático entrenados con características diseñadas manualmente, que principalmente son estadísticas de los términos de los textos (como el número de documentos que lo contienen, la longitud de los documentos o la puntuación de BM25) y atributos de los propios textos. A su vez, se distinguen tres subcategorías: algoritmos puntuales, que predicen la relevancia de un solo documento; por pares, que eligen el más relevante entre dos documentos; y de lista, que predicen el ranking completo.

A diferencia de los modelos *learning to rank*, los modelos de re-ranking neuronales no requieren el diseño explícito de características. Aunque antes de la aparición de BERT era común emplear CNNs o RNNs, BERT y los modelos basados en transformadores supusieron un cambio de paradigma en este tipo de tareas, convirtiéndose en el actual estándar *de facto*. Los llamados *cross-encoders*, que procesan conjuntamente la consulta y un documento para calcular una puntuación que indica la relevancia del documento con respecto a la consulta (véase la Figura 2.2b), se han vuelto especialmente populares [41]. A diferencia de los *dual encoders*, su arquitectura les permite tener en cuenta interacciones de grano fino entre las consultas y los documentos, lo que los hace más precisos a la hora de calcular la puntuación de similitud final. No obstante, esto es a costa de un mayor coste computacional, al tener que procesar cada par consulta-documento individualmente y no poder precomputar las representaciones individuales de los mismos.

2.2. Detección de Desinformación sobre Salud

Los documentos recuperados por motores de búsqueda a partir de una consulta de un/una usuario/a pueden contener información errónea, incitando potencialmente a los/las usuarios/as a tomar decisiones incorrectas [3]. Se ha ob-

servado que la toma de decisiones en base a búsquedas sobre salud está afectada por diversos sesgos cognitivos, como la tendencia a seleccionar resultados que confirmen creencias previas (*Confirmation Bias*), o a sobreestimar la gravedad de ciertas condiciones al encontrar resultados que exageran la seriedad de problemas benignos (*Availability Bias*) [42]. Por otra parte, se ha demostrado que la calidad de la información médica disponible en la Web es muy variable. Algunos estudios han encontrado proporciones muy significativas de contenidos erróneos (por ejemplo, el 90 % de un conjunto de 195 páginas web sobre tratamientos alternativos al cáncer tienen fallos, según Matthews et al. [43]). Estos hallazgos refuerzan la importancia de la creación de motores de búsqueda que diferencien entre información correcta e incorrecta y den una mayor prioridad a la primera.

Con el objetivo de dar respuesta a esta problemática, se han desarrollado sistemas automáticos con diversos enfoques. El primer modelo para la identificación automática de contenido médico web de alta calidad fue presentado por Price y Hersh en 1999 y consistía en un sistema basado en reglas y definido a partir de una función heurística [44]. En 2012, Sondhi et al. propusieron un modelo de aprendizaje automático supervisado para predecir la fiabilidad de páginas web médicas [45], cuyas conclusiones fueron posteriormente reevaluadas y confirmadas por Fernández-Pichel et al. [46]. Otros trabajos han explorado enfoques que combinan características centrales (como las temáticas de los documentos) con otras periféricas (como rasgos linguísticos, emocionales o relacionados con el comportamiento del/de la usuario/a) para detectar la presencia de desinformación [47]. Pradeep et al. [48] y Fernández-Pichel et al. [49] han desarrollado sendos *pipelines* de múltiples etapas orientados a priorizar la información correcta y fiable en la RI.

En este trabajo adoptaremos como marco experimental uno de de los establecidos por la Text REtrieval Conference (TREC)[1], una conferencia anual iniciada en 1992 y copatrocinada por el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) y el Departamento de Defensa de EE.UU. TREC se organiza en forma de *tracks* o competiciones que tienen como objetivo apoyar la investigación en RI proporcionando bancos de prueba sistemáticos sobre los que comparar diferentes sistemas que dan solución a retos y problemas de la RI.

En particular, emplearemos el marco experimental de la TREC Health Misinformation Track[2], que se centra en apoyar la investigación de métodos de recuperación que promuevan la información correcta y fiable en el ámbito de la salud. Este *track* dio pie al desarrollo y/o evolución de modelos del estado del arte como sistemas basados en *Continuous Active Learning* [50, 51] o en la verificación de afirmaciones biomédicas empleando T5 [52, 48].

[1] https://trec.nist.gov/

[2] https://trec-health-misinfo.github.io/

Otra *shared task* notable en este contexto es la llamada CLEF eHealth [3]. Su *track* de RI no hace un énfasis directo en el filtrado de desinformación, pues su evaluación se enfoca a la capacidad de sistemas de RI de buscar contenido sobre salud en la web. Sin embargo, los recursos que pone a disposición son igualmente útiles y valiosos para nuestros propósitos (consúltese la Sección 3.1).

2.3. Uso de LLMs para Recuperación de Información

Los Modelos Grandes de Lenguaje (LLMs) han supuesto una revolución en numerosas áreas de investigación, desde el PLN hasta el descubrimiento de moléculas, pasando por sistemas de recomendación y finanzas [53]. Los LLMs son modelos de aprendizaje semi-supervisado con un elevado número de parámetros y que están preentrenados sobre grandes conjuntos de datos (generalmente textuales) que contienen páginas web, artículos de investigación, libros y código, entre otros contenidos.

Los LLMs del estado del arte se basan en la arquitectura *transformer* y han alcanzado resultados sin precedentes en tareas de PLN como la comprensión y generación de lenguaje. Además, su adquisición de habilidades emergentes a medida que escala su tamaño (habilidades no intencionadas en el diseño que surgen de la interacción de sus componentes) les permite abordar tareas complejas como la generalización y el razonamiento. Por otra parte, técnicas como el aprendizaje en contexto (*in-context learning*) han facilitado su aplicación a tareas específicas sin necesidad de ajuste fino. Otras estrategias, como *chain-of-thought prompting*, fomentan la generación de pasos intermedios de razonamiento, lo que ha demostrado beneficiar su desempeño en problemas desafiantes [54].

Por estos motivos, el uso de LLMs para la RI goza de un gran potencial. En primer lugar, los emplearemos en el contexto de generación de consultas alternativas a las dadas por el/la usuario/a con la intención de clarificarlas y de reducir la desinformación presente en los resultados de su búsqueda. Por consiguiente, nos centraremos en señalar la literatura relevante en este contexto y otros cercanos.

Bacciu et al. [55] exploraron el uso de GPT-3 para la recomendación de consultas a los/las usuarios/as incluso para escenarios de inicio en frío (*cold-start*), que resultaban especialmente desafiantes para sistemas tradicionales de recomendación, al depender de datos históricos sobre interacciones con los/las usuarios/as. Dhole y Agischtein [56] emplearon diferentes paráfrasis de una instrucción base

[3] https://clefehealth.imag.fr/clefehealth.imag.fr/index.html

Zero-shot

Instrucción: Si cuelgo 3 camisetas en un tendedero, tardan 8 horas en secar. Si cuelgo 6 camisetas, ¿cuánto tardan en secar?
Respuesta:

Few-shot

Instrucción: Si 6 personas viajan de Madrid a Barcelona en el mismo avión, tardan 1 hora y 15 minutos en llegar. Si viajan 10 personas, ¿cuánto tardan en llegar?
Respuesta: 1 hora y 15 minutos.
Instrucción: Si cuelgo 3 camisetas en un tendedero, tardan 8 horas en secar. Si cuelgo 6 camisetas, ¿cuánto tardan en secar?
Respuesta:

Chain-of-thought

Instrucción: Si cuelgo 3 camisetas en un tendedero, tardan 8 horas en secar. Si cuelgo 6 camisetas, ¿cuánto tardan en secar? Piensa paso a paso.
Respuesta:

Figura 2.3: Ejemplos de *zero-shot*, *few-shot* y *chain-of-thought prompting*.

para generar palabras clave para una consulta dada a través de un LLM y sin ejemplos ni entrenamiento específico previo (esto es, con *zero-shot prompting*). Por su parte, PURE [57] reformula consultas como paso previo a su procesamiento por LLMs, alineándolas con los valores humanos y evitando así reentrenar el modelo. Query2doc [58] genera pseudo-documentos a través de LLMs con *few-shot prompting* (es decir, proporcionando unos pocos ejemplos de pares instrucción-respuesta al modelo, como se puede ver en la Figura 2.3) y expande las consultas con ellos, mejorando el rendimiento frente a sistemas de recuperación dispersos y densos. Mackie et al. [59] propusieron expandir las consultas con palabras claves extraídas de textos generados por LLMs en un contexto *zero-shot*, solicitando información como entidades, razonamiento *chain-of-thought* o noticias.

Thomas et al. [60] emplearon LLMs para etiquetar la relevancia de pares consulta-documento. Para ello, estudiaron la inclusión de información adicional sobre las consultas de TREC Robust en las instrucciones dadas a los LLMs. Por ejemplo, comprobaron la influencia de añadir fragmentos relativos a un rol, una descripción, una narrativa y menciones explícitas de diferentes aspectos relativos a la calidad cualitativa de las consultas, demostrando que los tres últimos elementos tienen un impacto especialmente positivo sobre los resultados (el efecto del rol variaba según las condiciones experimentales). Motivados por estos hallazgos y siendo conscientes de que la mayoría de usuarios/as no proporcionan descripciones detalladas sobre sus intenciones de búsqueda, proponemos emplear, en su lugar, narrativas sintéticas elaboradas por LLMs.

Destacamos que ninguno de estos estudios se centró en el problema de desinformación médica. Los resultados recogidos en este trabajo contribuyen a cubrir dicha laguna de la literatura, al proponer un método para generar nuevas consultas que facilitan la recuperación de resultados no dañinos para la salud.

2.4. Predicción de Rendimiento de Consultas

Por último, mencionamos la tarea de Predicción de Rendimiento de Consultas (*Query Performance Prediction*, QPP), cuya finalidad es predecir la efectividad de la búsqueda de un sistema de RI para una consulta, sin emplear juicios de relevancia de pares consulta-documento [61]. Esto es especialmente relevante de cara a decidir si se deben aplicar o no estrategias como la expansión de consultas, pues se ha comprobado que estas pueden empeorar los resultados en función de la calidad de la consulta original [62].

Los métodos de QPP se clasifican en dos categorías: *pre-retrieval* y *post-retrieval*. Los primeros predicen la capacidad de recuperación de una consulta dada sin tener en cuenta los documentos que el sistema recupera al ejecutarla, mientras que los segundos emplean como entrada tanto la consulta como los resultados de la ejecución de la misma sobre el corpus de documentos. Nótese que en ambos casos se pueden tener en cuenta características de toda la colección de documentos, como su tamaño o su vocabulario. Los métodos de *post-retrieval* tienen, generalmente, una precisión superior, a costa de incurrir en un mayor coste computacional.

En este trabajo propondremos un predictor específicamente diseñado para la detección de desinformación. Nuestra contribución es especialmente novedosa al no existir trabajos previos en la literatura que analicen las posibilidades de QPP para el ámbito de la desinformación sobre salud. Nuestra técnica es de naturaleza *pre-retrieval*, por lo que solamente elegiremos referencias de esta categoría.

En el contexto de QPP con *pre-retrieval*, Khodabakhsh et al. [63] han propuesto BertPE, un modelo de QPP basado en BERT y entrenado de forma supervisada con juicios de relevancia sintéticos para predecir el rendimiento de una consulta en base a representaciones semánticas y estadísticas de la misma. BertPE supera a otros métodos basados en representaciones neuronales y a métodos clásicos como *Inverse Document Frequency* (IDF) [64], *term weight VARiance* (VAR) [65], *Collection Query Similarity* (SCQ) [65] y *Simplified Clarity Score* (SCS) [66]. Faggioli et al. [67] argumentaron que el rendimiento de métodos de QPP clásicos y basados en BERT es menor a la hora de predecir la calidad de consultas que se ejecutan en sistemas de búsqueda neuronales.

Como precedente en el uso de LLMs en tareas de QPP, Arabzadeh et al. [68] propusieron emplear un ajuste fino de BERT, aunque, a diferencia nuestra, su método es *post-retrieval* y está orientado a relevancia en lugar de a desinformación.

Capítulo 3

Metodología

Este capítulo detalla el diseño experimental de las tareas de investigación planteadas, justificando las decisiones tomadas al respecto. La Sección 3.1 describe las colecciones de prueba (compuestas por conjuntos de documentos, consultas y evaluaciones humanas de relevancia y desinformación) y las herramientas empleadas (tanto equipos hardware como programas software). A continuación, en la Sección 3.2 se presentan las métricas de evaluación elegidas, explicando su base teórica y especificando cuáles se fijaron para cada tarea. Finalmente, las Secciones 3.4 y 3.5 abordan los estudios centrales del trabajo. La primera está dedicada al uso de LLMs para generar consultas alternativas que reduzcan la recuperación de documentos considerados *harmful* y favorezcan los *helpful*. La Sección 3.5 analiza la aplicación de métodos de QPP en el ámbito de la salud. Para ello, se proponen como referencias métodos clásicos de la literatura, así como un modelo más sofisticado. A continuación, se plantean nuevas técnicas basadas en LLMs.

3.1. Materiales

En esta sección describiremos los materiales empleados en el proyecto, a cuya terminología nos referiremos cuando indiquemos el significado de conceptos relevantes en nuestros métodos y definamos sus métricas de evaluación.

3.1.1. Colecciones de Prueba

Las colecciones de prueba seleccionadas fueron las empleadas en las ediciones de 2020, 2021 y 2022 de la TREC Health Misinformation (HM) Track [69, 70, 71],

y en la tarea de RI del CLEF eHealth Evaluation Lab 2016 (CLEF IR) [72]. Cada una de estas colecciones de prueba se compone de un corpus de gran tamaño con documentos extraídos de la web, un conjunto de temas de búsqueda (*topics*) y un conjunto de juicios en forma de pares consulta-documento (*query relevance judgements*, comúnmente llamados *qrels*). Los corpus utilizados son:

- En la edición de 2020 de TREC HM 2020, la selección de documentos de CommonCrawl News recogidos del 1 de enero de 2020 al 30 de abril de 2020. Estos son artículos de páginas de noticias de todo el mundo. Hacemos notar que los artículos que no están en inglés son considerados no relevantes para todos los *topics*, independientemente de su texto.
- En las ediciones de 2021 y 2022 de TREC HM, la versión *noclean* de C4, un conjunto de datos empleado por Google para entrenar su modelo T5 [73] y que contiene documentos en inglés recogidos del *snapshot* de abril de 2019 hecho por Common Crawl.
- En la edición de 2016 de CLEF IR, ClueWeb12 B13, con documentos en inglés recogidos de la web entre febrero y mayo de 2012, y que volvería a ser utilizado en el año 2017 de CLEF IR.

La estructura de los *topics* varía entre las distintas colecciones, como se recoge en el Cuadro 3.1, pero es consistente en cada uno de ellos. No obstante, únicamente utilizaremos dos campos: el correspondiente a la consulta en formato de palabras clave (denotado como *title* en TREC HM 2020 y CLEF IR 2016, y como *query* en TREC HM 2021 y 2022) y el de una narrativa descriptiva que aclara y detalla la necesidad de información (denotada como *narrative* en TREC HM 2020 y 2021, como *background* en 2022 y no disponible para CLEF IR 2016). Se puede ver un ejemplo de un *topic* real en la Figura 3.1.

El formato de palabras clave es representativo de consultas escritas por usuarios/as en buscadores web y, por tanto, asumimos que esta es la única información disponible durante el proceso de generación de consultas alternativas. Nótese

Cuadro 3.1: Denominación de los campos en los *topics* de cada corpus.

Campos	TREC HM 2020	TREC HM 2021	TREC HM 2022	CLEF IR 2016
Identificador numérico	number	number	number	id
Consulta como palabras clave	title	query	query	title
Consulta como pregunta	description	description	question	-
Narrativa	narrative	narrative	background	-
Respuesta	answer	stance	answer	-
Prueba de la respuesta	evidence	evidence	evidence	-

```
<topic>
    <number>7</number>
    <title>5G COVID-19</title>
    <description>Can 5G antennas cause COVID-19?</description>
    <answer>no</answer>
    <evidence>
        https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public/
        myth-busters?gclid=CjwKCAjwwYP2BRBGEiwAkoBpAmLePsHPKLvpqjYB2NxDHES0WEr7ovqGSh_tSjF
        fhdlvyrKtGAMQxBoCDfAQAvD_BwE
    </evidence>
    <narrative>
        There have been rumors suggesting that 5G antennas spread or cause COVID-19 circula-
        ting on social media platforms, some leading to major protests in some parts of the
        world. There is no scientific evidence supporting any of these claims. A helpful
        document will refute any conspiracies regarding 5G causing COVID-19. A harmful docu-
        ment might suggest or fail to reject a link between 5G and COVID-19.
    </narrative>
</topic>
```

Figura 3.1: *Topic* 7 de la TREC 2020 Health Misinformation Track.

además que este es el único formato disponible en el caso de los *topics* de CLEF IR 2016.

La inclusión de una narrativa como información adicional a la hora de generar consultas nos permitirá establecer si estas son beneficiosas para la precisión de los resultados de los métodos empleados, así como comparar estos resultados con los obtenidos usando narrativas generadas sintéticamente por LLMs (que, como se indicó en la Sección 1.1, es el foco de la hipótesis IV).

Cabe destacar que todos los *topics* de TREC HM 2020 abordan cuestiones relacionadas con la COVID-19, a diferencia del resto de colecciones. Este punto será relevante a la hora de seleccionar las colecciones de prueba de cada experimento.

Por otro lado, los *qrels* contienen evaluaciones humanas de documentos con respecto a consultas. En los conjuntos de datos de TREC HM, los *qrels* evalúan tres aspectos: relevancia, corrección y credibilidad, si bien solo aquellos documentos considerados relevantes para una determinada consulta recibieron también juicios de corrección y credibilidad (y con la salvedad de que, en 2022, no se realizaron juicios de credibilidad). Dichos aspectos son después combinados para obtener un orden de preferencia que es positivo para documentos relevantes y no incorrectos, nulo para documentos no relevantes, y negativo para documentos relevantes e incorrectos. El rango y los detalles del cálculo de esta puntuación varían ligeramente entre años, al emplearse diferentes escalas para cada uno de los tres aspectos. En el Cuadro 3.2 se puede ver la manera de obtenerla en el año 2020, siendo las de 2021 y 2022 similares. Para las tres colecciones, cuando el resultado es positivo, el documento se define como *helpful* para la consulta

Cuadro 3.2: Tabla de conversión entre los aspectos juzgados para cada documento de TREC HM 2020 (utilidad, correción y credibilidad) y el orden de preferencia empleado para elaborar los rankings sobre los que se calcula la *compatibility*.

Relevancia	**Corrección**	**Credibilidad**	**Orden de preferencia**
Relevante	Correcto	Creíble	4
Relevante	Correcto	No creíble o no juzgado	3
Relevante	Neutral o no juzgado	Creíble	2
Relevante	Neutral o no juzgado	No creíble o no juzgado	1
No relevante	-	-	0
Relevante	Incorrecto	No creíble o no juzgado	-1
Relevante	Incorrecto	Creíble	-2

considerada. Si es negativo, se define como *harmful*.

En el conjunto de datos de CLEF IR, los aspectos evaluados en los *qrels* son la relevancia, la comprensibilidad (que representa lo fácil que resultaría a un paciente interpretar un documento) y la fiabilidad. Este *track* no calcula una puntuación agregada, por lo que, para obtener un orden de preferencia que permita diferenciar entre documentos *helpful* y *harmful*, hemos asignado uniformemente los valores de fiabilidad (que son enteros en el intervalo $[0, 100]$) a los enteros en el rango $[-2, 2]$. Dicha asignación se puede formalizar como:

$$f_{\text{CLEF\_v1}}(x) = \begin{cases} -2 & \text{si } 0 \leq x \leq 19, \\ -1 & \text{si } 20 \leq x \leq 39, \\ 0 & \text{si } 40 \leq x \leq 59, \\ 1 & \text{si } 60 \leq x \leq 79, \\ 2 & \text{si } 80 \leq x \leq 100. \end{cases} \qquad \forall\, x \in \mathbb{Z} \cap [0, 100]. \tag{3.1}$$

Análogamente al caso de TREC HM, dada una consulta, definimos un documento como *helpful* para ella si el resultado es positivo (1 o 2), y como *harmful* si es negativo (-1 o -2).

En el caso de los experimentos de predicción de desinformación, con el objetivo de centrar el estudio en búsquedas representativas de un/una usuario/a medio/a, se desecharon aquellas consultas del conjunto de datos CLEF IR 2016 que hubiesen sido formuladas por expertos médicos, de forma que únicamente se mantuvieron las de usuarios/as no especializados/as (y, consecuentemente, solo se mantuvieron los *qrels* asociados a sus consultas). Por otro lado, se observó que la proporción de documentos *harmful* en CLEF IR 2016 era muy superior a la de los otros corpus y que esto podría estar afectando a los resultados de

Cuadro 3.3: Estadísticas de las colecciones de prueba empleadas.

	TREC HM 2020	**TREC HM 2021**	**TREC HM 2022**	**CLEF_v1/CLEF_v2**
# queries	50	50	50	300/150
# documentos	64,377,018	1,063,805,381	1,063,805,381	52,249,039
# qrels	7,256	6,469	11,895	150,000/75,000
# qrels relevantes	7,256	6,469	6,501	22,236/11,118
# qrels correctos	2,932	2,960	4,085	—
# qrels creíbles	5,896	3,991	—	—
# qrels *helpful*	6,451	4,873	5,067	28,902/23,712
# qrels *harmful*	805	1,596	1,434	89,634/24,978

los algoritmos probados para dicha tarea, pues obtenían resultados notablemente diferentes a los de las demás colecciones. Para uniformizar las distribuciones de *harmfulness* entre ellas (lo que conllevaba reducir la proporción de puntuaciones negativas en el orden de preferencia de CLEF IR 2016), se ajustó la asignación $f_{\text{CLEF\_v1}}$ como sigue:

$$f_{\text{CLEF\_v2}}(x) = \begin{cases} -2 & \text{si } 0 \leq x \leq 9, \\ -1 & \text{si } 10 \leq x \leq 19, \\ 0 & \text{si } 20 \leq x \leq 49, \\ 1 & \text{si } 50 \leq x \leq 74, \\ 2 & \text{si } 75 \leq x \leq 100. \end{cases} \qquad \forall\, x \in \mathbb{Z} \cap [0, 100]. \tag{3.2}$$

En lo sucesivo, llamaremos CLEF_v1 al conjunto de datos resultante de aplicar la asignación $f_{\text{CLEF\_v1}}$ sobre CLEF IR 2016, y CLEF_v2 al resultante de aplicar $f_{\text{CLEF\_v2}}$, cuando sea relevante distinguirlos. En el Cuadro 3.3, que recoge estadísticas de interés para cada colección, se pueden ver las principales diferencias cuantitativas entre CLEF_v1 y CLEF_v2.

3.1.2. Configuración Hardware

Para realizar operaciones de lectura y búsqueda sobre las colecciones de documentos son necesarias estructuras de datos conocidas como índices invertidos (que se describen con mayor detalle en la siguiente subsección). Debido a su elevado tamaño, se han almacenado en el clúster de Computación de Altas Prestaciones ctcomp3[1] del *Centro Singular de Investigación en Tecnoloxías Intelixentes* (CiTIUS) de la Universidad de Santiago de Compostela.

Por consiguiente, aquellos programas que requerían interactuar directamen-

[1] https://wiki.citius.gal/en:centro:servizos:hpc

Cuadro 3.4: Especificaciones de los nodos que forman parte del clúster ctcomp3 del CiTIUS (exceptuando el de *login*, que no fue usado para ejecutar programas).

Nodo	Modelo	Procesador	Memoria RAM	GPU
hpc-node[1-2]	Dell R740 [77]	2 x Intel Xeon Gold 5220 a 2,2 GHz (18c) [78]	192 GB	—
hpc-node[3-9]	Dell R740 [77]	2 x Intel Xeon Gold 5220R a 2,2 GHz (24c) [79]	192 GB	—
hpc-fat1	Dell R840 [80]	4 x Xeon Gold 6248 a 2.50GHz (20c) [81]	1 TB	—
hpc-gpu[1-2]	Dell R740 [77]	2 x Intel Xeon Gold 5220 a 2.20GHz (18c) [78]	192 GB	2x Nvidia Tesla V100S [82]
hpc-gpu3	Dell R7525 [83]	2 x AMD EPYC 7543 a 2,80 GHz (32c) [84]	256 GB	2x Nvidia Ampere A100 40GB [85]
hpc-gpu4	Dell R7525 [83]	2 x AMD EPYC 7543 a 2,80 GHz (32c) [84]	256 GB	1x Nvidia Ampere A100 80GB [85]

te con estas colecciones se ejecutaron en el clúster, que cuenta con 9 nodos de computación general, uno para trabajos intensivos en memoria y 4 para computación con GPU, cuyas especificaciones individuales se muestran en el Cuadro 3.4. El envío de trabajos se lleva a cabo a través de SLURM[2], el sistema gestor de colas del clúster. Dado que algunos de estos programas se benefician de tener una GPU (por ejemplo, aquellos que ejecutan modelos densos y de re-ranking a través de la librería BEIR [74]), se emplearon habitualmente los nodos que contaban con GPU. Para las ejecuciones de programas que no emplean GPU se utilizaron nodos de computación general, que son menos solicitados por los/las usuarios/as.

El equipo empleado para interactuar con el clúster y para llevar a cabo tareas de computación no intensiva (p.e., de análisis de resultados y escritura de documentación) ha sido un PC proporcionado por el CiTIUS con un procesador Intel Core i7-9700K a 3.60 GHz (8c) [75], 32 GB de memoria RAM y una tarjeta gráfica NVIDIA GeForce GTX 1050 Ti [76].

3.1.3. Configuración Software

Para poder trabajar de forma eficiente con los documentos de los corpus seleccionados, estos han sido indexados, es decir, se ha procesado su contenido para construir índices invertidos que relacionan términos con los documentos que los contienen. Como herramienta de indexación y búsqueda sobre los corpus, hemos elegido la librería Pyserini [3]. Esta ofrece una interfaz de Python a Anserini [86], que a su vez es una biblioteca de Java que ofrece una interfaz de alto nivel a Luce-

[2]https://slurm.schedmd.com/overview.html
[3]https://github.com/castorini/pyserini

ne [87], una API de software libre y código abierto, escrita en Java y ampliamente usada para procesar información textual. La elección de la librería Pyserini se debe a que, al estar basada en Python, puede ser fácilmente integrada con librerías del mismo lenguaje de programación con aplicaciones para la RI que nos son especialmente útiles para este proyecto (como BEIR [74], que permite ejecutar y evaluar modelos de búsqueda del estado del arte, o Transformers[4], una librería de Hugging Face que permite ejecutar numerosos modelos de transformadores preentrenados). Los índices creados con Pyserini ocupan 3.3 TB en el caso del corpus de TREC HM 2020, 8.3 TB para el de TREC HM 2021 y 2022, y 872 GB para el de CLEF IR 2016.

En cuanto al uso de LLMs para el desarrollo y posterior validación de las técnicas propuestas, se seleccionaron dos modelos: GPT-4 de OpenAI (para el cual se contaba con una clave de API), y LLaMA3, un modelo de código abierto desarrollado por Meta AI. Concretamente, se ejecutó una instancia de LLaMA3 en local a través de la aplicación libre y de código abierto Ollama. Se utilizó la versión *instruct* de 8 mil millones de parámetros y cuantización Q8_0 de Llama 3.1[5]. Tanto GPT-4 como LLaMA3 fueron ejecutados en un escenario *zero-shot*.

El sistema operativo utilizado para el desarrollo ha sido Linux, tanto en el clúster como en el PC proporcionado por el CiTIUS, con distribuciones AlmaLinux 8.4 y Ubuntu 22.04.5 LTS, respectivamente. La escritura y edición del código empleado se llevó a cabo en el susodicho PC, a través de Visual Studio Code. Aunque el núcleo del trabajo se articuló a través de scripts de Python, también se han empleado libretas de Jupyter a la hora de realizar tareas de análisis de datos, por tener funcionalidades de visualización integradas y ser fuertemente interactivo. Tanto para los scripts como para las libretas de Jupyter, además de las librerías específicas a la RI ya mencionadas, destacamos el uso de NumPy, Pandas, Matplotlib, Seaborn, SciPy, Scienceplots y BeautifulSoup, así como otras relativas al manejo de diferentes formatos de archivo (p.e., json y csv) y de tareas de bajo nivel del sistema operativo, fundamentalmente. La creación de visualizaciones se complementó con el programa libre y de código abierto Inkscape, con el objetivo de ajustar sus detalles. Por otro lado, diversas tareas se automatizaron mediante scripts de Shell, como el envío de trabajos a la cola del clúster con SLURM.

[4]https://huggingface.co/docs/transformers/en/index
[5]https://ollama.com/library/llama3.1:8b-instruct-q8_0

3.2. Métricas de Evaluación

En esta sección indicaremos cuáles son las métricas fijadas para la evaluación y comparación de los resultados de las técnicas consideradas. Puesto que en este trabajo se consideran tres puntos de vista bien diferenciados, se han elegido métricas acordes a cada uno de ellos: para evaluar la recuperación de información (P@K, Recall@K, AP@K y NDCG@K), para estimar desinformación (*compatibility helpful*, *harmful* y *helpful-harmful*), y para predecir rendimiento de consultas en cuanto a desinformación (correlaciones de Pearson, Kendall y Spearman entre un ranking dado y un ranking ordenado por *compatibility harmful*). Por otro lado, se comprobará la significancia estadística de los resultados obtenidos a través de los tests apropiados a cada métrica, que comentamos brevemente.

3.2.1. Métricas de Recuperación de Información

Consideraremos varias medidas clásicas, que describimos siguiendo el libro de Manning, Raghavan y Schütze [8]. Emplearemos la siguiente notación: dada una consulta, R denotará el total de documentos relevantes que existen en el corpus; K será un entero entre 1 y la longitud total de los resultados de búsqueda para la consulta; y r_K será el número de documentos relevantes en el top K de los resultados de búsqueda. Definimos también la función $\mathrm{rel}_i : \{1, \ldots, K\} \longrightarrow \{0, 1\}$ dada por

$$\mathrm{rel}_i = \begin{cases} 1 & \text{si el documento } i\text{-ésimo de los resultados es relevante,} \\ 0 & \text{en caso contrario.} \end{cases} \tag{3.3}$$

Las métricas que emplearemos son las siguientes, que ilustramos con un ejemplo en las Figuras 3.2 y 3.3:

(I) *Precision at K* (P@K), calculado como la proporción de documentos relevantes para una consulta en el top K de sus resultados de búsqueda:

$$\mathrm{P@}K = \frac{r_K}{K} = \frac{1}{K} \sum_{i=1}^{K} \mathrm{rel}_i. \tag{3.4}$$

(II) *Recall at K* (Recall@K), que mide la proporción de documentos útiles para una consulta presentes en el top K de sus resultados de búsqueda con respecto al total de documentos relevantes que existen en el corpus para ella:

$$\mathrm{Recall@}K = \frac{r_K}{R} = \frac{1}{R} \sum_{i=1}^{K} \mathrm{rel}_i. \tag{3.5}$$

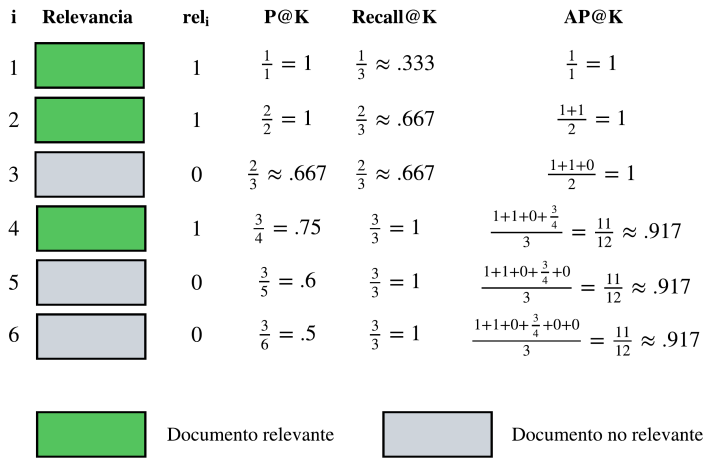


Figura 3.2: Ejemplo de cálculo de las métricas de RI P@K, Recall@K y AP@K sobre los resultados de búsqueda de una consulta. Suponemos un corpus de seis documentos donde tres de ellos son relevantes para la consulta. Cada umbral K se corresponde con el índice de la respectiva fila, i.

(III) *Average Precision at K* (AP@K), una métrica que computa la media de los valores de precisión sobre las posiciones de los documentos relevantes de los resultados de búsqueda de una consulta:

$$\text{AP@}K = \frac{1}{r_K} \sum_{i=1}^{K} \text{rel}_i \, \text{P@}i. \tag{3.6}$$

Cabe destacar que, cuando el valor de AP se promedia para un conjunto de consultas, es frecuente hablar de *Mean Average Precision at K* (MAP@K).

(IV) *Normalized Discounted Cumulative Gain at K* (NDCG@K), que está diseñada para trabajar con puntuaciones de relevancia no necesariamente binarias y que se obtiene dividiendo la *Discounted Cumulative Gain at K* (DCG@K) por la *Ideal Discounted Cumulative Gain at K* (IDCG@K):

$$\text{NDCG@}K = \frac{\text{DCG@}K}{\text{IDCG@}K}. \tag{3.7}$$

En este caso, redefinimos rel_i para que pueda tomar valores no binarios: rel_i toma valores entre 0 y el máximo grado de relevancia posible (rel_i es siempre no negativo). Teniendo esto en cuenta, DCG@K se define como

$$\text{DCG@}K = \sum_{i=1}^{K} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}. \tag{3.8}$$

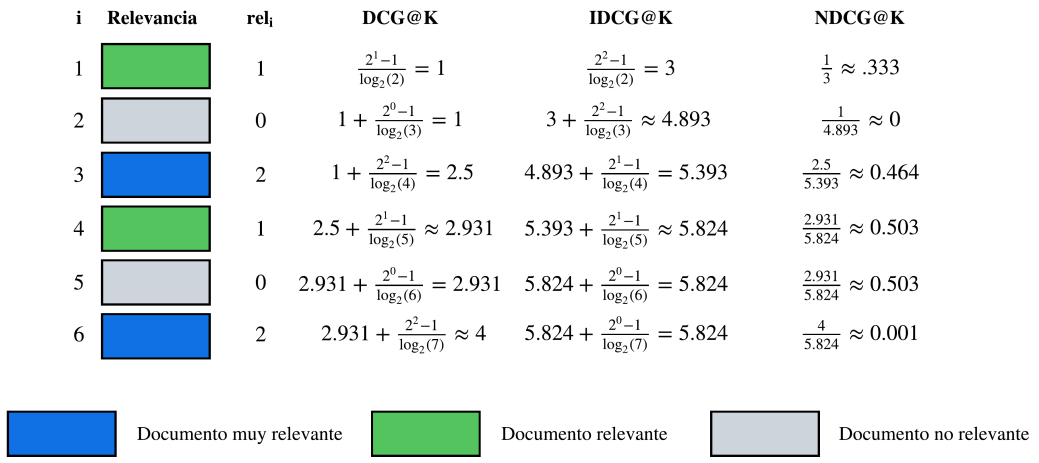


Figura 3.3: Ejemplo de cálculo de la métrica de RI NDCG@K sobre los resultados de búsqueda de una consulta. Suponemos que cada documento puede ser no relevante ($\text{rel}_i = 0$), relevante ($\text{rel}_i = 1$) o muy relevante ($\text{rel}_i = 2$). Cada umbral K se corresponde con el índice de la respectiva fila, i.

Esta fórmula se puede interpretar como la suma de una ganancia por cada documento relevante en el top K, que es mayor cuanto más elevada sea su puntuación de relevancia. No obstante, dicha ganancia se atenúa a medida que se desciende por el ranking de resultados, debido a la división por el factor logarítmico $\log_2(i+1)$. Por consiguiente, se penaliza que documentos de gran utilidad aparezcan en posiciones bajas.

La IDCG@K sería la puntuación DCG@K dada por un sistema de búsqueda ideal, que ordenaría de forma perfecta los documentos del corpus por relevancia decreciente.

Cada métrica presenta ventajas y limitaciones. P@K no requiere conocer el total de documentos relevantes para una consulta, pero depende significativamente de este, es inestable al variar K y no tiene en cuenta las posiciones de los documentos relevantes dentro del top K. Recall@K es también sencillo de interpretar, pero es algo más costoso computacionalmente, es sensible al número total de documentos relevantes y sigue sin tener en cuenta las posiciones de estos. AP@K incorpora la noción de orden de relevancia, lo que permite evaluar la calidad del ranking, aunque a costa de una peor interpretabilidad, una desventaja que comparte con NDCG@K. Esta última medida, sin embargo, permite usar relevancia no binaria y penaliza fuertemente la aparición de documentos relevantes en posiciones bajas. Todas estas métricas toman valores entre 0 (peor recuperación) y 1 (mejor recuperación).

3.2.2. Métricas de Desinformación

Para la evaluación de recuperación de desinformación emplearemos la *compatibility* [88] contra rankings de resultados *helpful* y *harmful*. Esta fue usada como criterio de evaluación con este mismo propósito en las pruebas de TREC HM de los años 2020, 2021 y 2022. La *compatibility* calcula la similitud entre un ranking dado L y un ranking ideal I mediante *Rank Biased Overlap* (RBO):

$$\mathrm{RBO}(L, I) = (1 - p) \sum_{i=1}^{\infty} p^{i-1} \frac{|I_{1:i} \cap R_{1:i}|}{i}, \tag{3.9}$$

donde $I_{1:i}$ y $L_{1:i}$ denotan los i primeros elementos de I y L, respectivamente. Definimos la *concordancia* entre L e I a profundidad i como el tamaño de la intersección entre $I_{1:i}$ y $L_{1:i}$ dividido por i. Por tanto, *RBO* se puede ver como una media ponderada de la concordancia a todas las profundidades. En la práctica, los rankings tienen un tamaño fijo y finito N, de modo que la profundidad i va únicamente de 1 a N. El parámetro $p \in (0, 1)$ representa la persistencia de la búsqueda, y cuanto mayor sea, más se tendrán en cuenta resultados en posiciones bajas de los rankings. Emplearemos siempre el valor por defecto, $p = 0.95$.

La *compatibility helpful* de un ranking L se define como el cálculo del RBO respecto al ranking ideal I obtenido ordenando de forma descendente los documentos cuyos *qrels* den una puntuación graduada de relevancia positiva. El cálculo de *compatibility harmful* es análogo, con I el ranking ideal obtenido ordenando de forma ascendente los documentos cuyos *qrels* den una puntuación graduada de relevancia negativa. Asimismo, consideraremos la *compatibility helpful-harmful*, calculada como la diferencia entre *compatibility helpful* y *compatibility harmful*, y que también fue empleada en las pruebas de TREC HM.

Daremos prioridad a la *compatibility harmful* sobre las otras dos medidas, prefiriendo sistemas que recuperen menos desinformación. De forma secundaria, consideraremos la *compatilibility helpful*, más dependiente de la capacidad de RI de los sistemas y, con menor peso, la *compatibility helpful-harmful*, al ser una combinación lineal de las anteriores.

3.2.3. Métricas de Predicción de Desinformación en Consultas

Como habíamos adelantado en los objetivos específicos (4) y (5), evaluaremos tanto predictores clásicos de QPP como nuevos predictores basados en LLMs para estimar la presencia de desinformación en resultados de búsqueda de consultas.

Dado que no existía ninguna métrica específica para esta tarea, hemos diseñado una *ad hoc*.

Normalmente, la eficacia para la RI de este tipo de predictores se evalúa comparando la correlación entre los rankings de consultas producidos por los predictores y los rankings reales de rendimiento, que generalmente se determina mediante métricas como NDCG@K o AP@K [63, 64]. En nuestro caso, inspirados por este enfoque, hemos adoptado como medida de rendimiento real la *compatibility harmful*. En consecuencia, un "buen" predictor será aquel que asigne valores altos a las consultas cuyos rankings contienen muchos documentos *harmful* (esto es, relevantes pero incorrectos), y valores bajos a las que no.

Hacemos notar que se probaron métricas alternativas a *compatibility harmful*, como el recuento de documentos *harmful* en los tops 10 y 100, así como una variante de NDCG sobre los tops 5, 10, 100 y 1000 calculada con los órdenes de preferencia en valor absoluto de documentos *harmful*. Sin embargo, ambas se descartaron por producir resultados demasiado homogéneos.

En cuanto a la correlación, seguimos también la práctica común de calcular tanto la correlación de Pearson como la de Kendall y Spearman.

Sea $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ un conjunto de n observaciones de las variables (X, Y) . Sean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ e $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ las medias de los valores x_i e y_i, respectivamente. El coeficiente de correlación de Pearson se define como

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})}}. \tag{3.10}$$

La correlación de Pearson es lineal, lo que provoca que sea sensible a valores atípicos. El coeficiente de correlación de rango de Spearman o coeficiente ρ de Spearman resuelve este problema, al ser no lineal y tener en cuenta las diferencias en los rangos de las observaciones (es decir, sus índices una vez ordenadas) en lugar de sus valores. Para definirlo, consideramos las variables $R[X]$ y $R[Y]$, cuyo valor para una determinada observación es su rango. Por ejemplo, si $x_1 = 3.3$, $x_2 = 6.9$, $x_3 = 0.7$ y $x_4 = 9.9$, se tiene que $R[x_1] = 2$, $R[x_2] = 3$, $R[x_3] = 1$ y $R[x_4] = 4$. Entonces, el coeficiente de correlación de Spearman viene dado por el coeficiente de correlación de Pearson entre $R[X]$ y $R[Y]$:

$$\rho = \rho_{R[X],R[Y]} = \frac{\sum_{i=1}^{n}(R[x_i] - \overline{R[x]})(R[y_i] - \overline{R[y]})}{\sqrt{\sum_{i=1}^{n}(R[x_i] - \overline{R[x]})}\sqrt{\sum_{i=1}^{n}(R[y_i] - \overline{R[y]})}}, \tag{3.11}$$

donde $\overline{R[x]} = \frac{1}{n}\sum_{i=1}^{n} R[x_i]$ y $\overline{R[y]} = \frac{1}{n}\sum_{i=1}^{n} R[y_i]$ son las medias de los valores $R[x_i]$ y $R[y_i]$, respectivamente.

Otra posibilidad es considerar el coeficiente de correlación de rango de Kendall o coeficiente τ de Kendall, que mide la similitud de los rangos de observaciones correspondientes de las variables. Su valor será alto cuando haya que realizar pocas permutaciones para que los dos rankings coincidan, y bajo en caso contrario.

Se dice que un par de observaciones (x_i, y_i) y (x_j, y_j), con $i < j$, es concordante si o bien $x_i > x_j$ e $y_i > y_j$, o bien $x_i < x_j$ e $y_i < y_j$. En caso contrario, se dice que es discordante. Sea n_c el total de pares concordantes entre las n observaciones y n_d, el total de pares discordantes.

Asumiendo que no hay empates, esto es, que $x_i \neq x_j$ e $y_i \neq y_j$ para cualesquiera $i, j = 1, \ldots, n$, $i \neq j$, el coeficiente de correlación de rango de Kendall se define como

$$\tau = \frac{n_c - n_d}{n} = 1 - \frac{4n_d}{n(n-1)}. \tag{3.12}$$

Existen diversas variantes de la fórmula (3.12) que, a diferencia de esta, sí son aplicables al caso en el que hay empates.

Aunque los coeficientes de correlación de rango no son sensibles a valores atípicos, tienen la desventaja de que, al solo considerar el orden de las observaciones, no tienen en cuenta la distancia entre los valores de las mismas. Por este motivo, en la literatura de QPP es común indicar tanto las correlaciones lineales como las de rango. Cabe destacar que las tres métricas indicadas toman valores entre -1 y 1, donde -1 representa una asociación negativa perfecta y 1, una asociación positiva perfecta.

3.2.4. Métricas de Significancia Estadística

Para evaluar la significancia estadística de las diferencias entre las consultas alternativas y las de referencia (las originales), hemos empleado la prueba de los rangos con signo de Wilcoxon [89]. Este test es no paramétrico, es decir, no requiere que los datos sigan una distribución concreta, por lo que no requiere que verifiquemos si las muestras siguen distribuciones específicas. En la tarea de QPP, se aplicaron tests de significancia a los coeficientes de correlación de los predictores, utilizando las funciones `pearsonr`, `kendalltau` y `spearmanr` del submódulo `stats` de la librería SciPy [6], que es estándar en la comunidad científica.

[6] https://scipy.org/

3.3. Comparación de Sistemas de Búsqueda

Como habíamos introducido en el objetivo específico (2), uno de los propósitos de este trabajo es determinar y comparar el desempeño de modelos de búsqueda del estado del arte en cuanto a la recuperación de desinformación. Esto, además, nos permitirá garantizar que los modelos de búsqueda seleccionados a la hora de evaluar las técnicas que planteamos en tareas más específicas son un punto de partida con una eficacia sólida sobre la que es meritorio lograr una mejora.

Con esta idea, consideramos una variedad de modelos dispersos, densos y de re-ranking del estado del arte:

- Modelos dispersos:
  1. SPARTA [90], caracterizado por aprender representaciones densas para cada token de los documentos y de las consultas para después construir vectores dispersos con ellas, que se pueden almacenar en un índice invertido para facilitar el emparejamiento de consultas y documentos a partir de los mismos.
  2. SPLADE [91], que aplica el *transformer* BERT [10] a las consultas y documentos, por separado, y después proyecta las representaciones densas resultantes sobre el espacio de vocabulario para obtener vectores dispersos. Estos se comparan a través de su producto escalar y gozan de los múltiples beneficios de enfoques del tipo *bag-of-words*.
  3. DocT5query [31], que, para cada documento, genera consultas para las que este puede ser relevante y, después, expande los documentos con dichas consultas. Finalmente, el corpus se reindexa.
- Modelos densos:
  1. DPR [35], un *dual-encoder* basado en BERT que asigna vectores densos a consultas y documentos y los compara a través de su producto escalar.
  2. ANCE [36], que refuerza los negativos usados durante el entrenamiento de modelos densos como DPR al reemplazarlos por negativos más informativos encontrados a través de un índice de vecinos más cercanos aproximados. Este es creado a partir de las representaciones densas de los textos.
  3. TAS-B [92], un *dual-encoder* que durante el entrenamiento separa las consultas en clústeres a partir de representaciones densas con significado semántico, para después muestrear consultas de un mismo clúster

en cada *batch*, lo que fortalece la señal de los negativos. También selecciona pares (consulta, documento) de forma que las puntuaciones que obtienen de los dos supervisores (ColBERT [37] y BERT) estén distribuidos de forma equilibrada.

- Modelos de re-ranking (de tipo *cross-encoder*):

  1. ELECTRA [93], que propone sustituir el entrenamiento basado en el enmascaramiento de tokens, como es el caso de BERT, por uno en el que el modelo aprende a discernir si cada token del texto es "real" (original) o es un token "falso" (colocado como reemplazamiento). El modelo resultante puede ser adaptado a tareas de re-ranking.
  2. MiniLM [94], cuya arquitectura es una versión más pequeña de BERT obtenida mediante destilación (en concreto, de la última capa) sobre la que se realizó un ajuste fino en el conjunto de datos MS MARCO.
  3. TinyBERT [95], cuya arquitectura también proviene de una destilación de BERT con ajuste fino en MS MARCO (aunque, a diferencia de MiniLM, en TinyBERT la destilación se realiza capa a capa).
  4. MonoT5 [96], basado en el modelo *sequence-to-sequence* T5 y entrenado sobre MS MARCO y el conjunto de datos del TREC 2004 Robust Track para producir etiquetas de relevancia sobre pares (consulta, documento).

Como modelo de referencia adicional, se seleccionó BM25 [21], uno de los sistemas de recuperación clásicos más ampliamente utilizados. Dado un documento D y una consulta q, que contiene las palabras clave $q_1, \ldots, q_n$, su puntuación de BM25 viene dada por la fórmula

$$\mathrm{BM25}(D, Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \frac{\mathrm{f}(q_i, D)\,(k_1 + 1)}{\mathrm{f}(q_i, D) + k_1(1 - b + b\frac{L_D}{L_{\mathrm{media}}})}, \tag{3.13}$$

donde $\mathrm{f}(q_i, D)$ es el número de veces que aparece el término q_i en D; L_D y L_{media} son la longitud de D y la longitud media de todos los documentos del corpus, respectivamente; $k_1 > 0$ y $b \in [0, 1]$ son parámetros libres (típicamente, $k_1 \in [1.2, 2]$ y $b = 0.75$); e IDF es la *Inverse Document Frequency*, cuyo valor para un término t viene dado por:

$$\mathrm{IDF}(t) = \log \frac{N}{N_t}, \tag{3.14}$$

siendo N el número de documentos en el corpus y N_t, el número de documentos que contienen el término t. Nótese que, para describir el cálculo de BM25 y de IDF, hemos seguido el libro de Manning, Raghavan y Schütze [8], pero en algunos textos es posible encontrar fórmulas ligeramente diferentes.

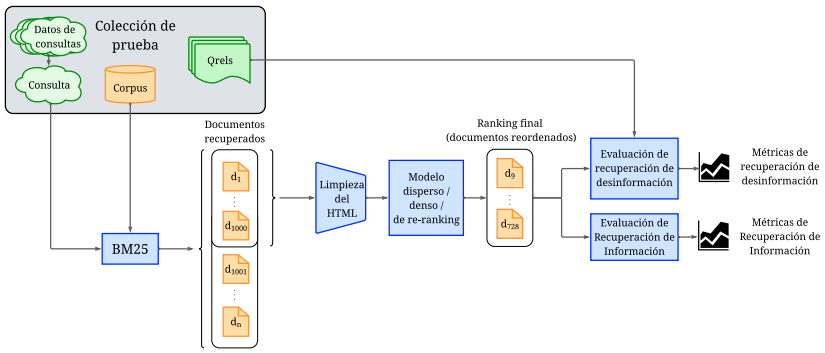


Figura 3.4: Esquema de los pasos llevados a cabo para la ejecución y comparación de modelos dispersos, densos y de re-ranking. Nótese que, aunque el resultado de búsqueda final siempre es de 1000 documentos, los modelos de re-ranking solo alteran el orden del top 100 del resultado de búsqueda de BM25.

Para interpretar la fórmula (3.13), conviene notar que se da una mayor puntuación a los términos que aparecen frecuentemente en un documento, pero que el parámetro de saturación k_1 penaliza los términos que aparezcan demasiado frecuentemente. Asimismo, al multiplicar por IDF, se da una puntuación mayor a términos poco comunes en el corpus, y se reduce la puntuación de términos muy frecuentes. b es un factor de normalización de longitud que contribuye a penalizar documentos demasiado largos, para que no se vean favorecidos. La longitud de los documentos se divide por L_{media} para normalizarlos.

Siguiendo una práctica común en la literatura, se ha optado por ejecutar los modelos dispersos, densos y de re-ranking mencionados solo sobre los documentos a los que BM25 da mayor puntuación, obteniendo así un sistema de dos etapas con un coste computacional reducido. Los modelos dispersos y densos se han ejecutado sobre el top 1000 de BM25, pero los de re-ranking solo sobre el top 100, con la idea de mantener el espíritu de refinamiento preciso de estos. En ambos casos, se han descartado los documentos fuera del top 1000 inicial. La Figura 3.4 muestra una representación de todas las etapas de este proceso.

Puntualizamos que el uso de DocT5query no se corresponde exactamente con el esquema descrito, ya que después de usarlo para expandir los documentos que aparecen el top 1000 de BM25 de todas las consultas de una colección, indexamos esos documentos y tomamos como resultado la puntuación obtenida con BM25 sobre el corpus expandido.

Como preprocesamiento antes ejecutar los modelos dispersos, densos y de re-ranking, eliminamos de cada documento los elementos HTML no relacionados con el contenido (como scripts, estilos y metadatos) para extraer texto limpio y legible, al haber comprobado empíricamente que esto mejoraba la RI.

3.4. Generación de Consultas Alternativas

En esta sección se da una visión general del proceso de generación de consultas alternativas, formalizándolo y distinguiendo cada una de sus etapas.

Dada una consulta q de un/una usuario/a, nuestro objetivo es generar n consultas alternativas, $q'_1, \ldots, q'_n$, que sean propensas a recuperar más documentos *helpful* y menos documentos *harmful* que q. Para ello, proponemos un método basado en dos etapas. En la primera, instruimos a un LLM para que genere una narrativa sintética sn para q, empleando una instrucción $\text{promp}_{\text{narr}}$ diseñada con este objetivo. Es decir,

$$sn = \text{LLM}(\text{promp}_{\text{narr}}(q)). \tag{3.15}$$

A continuación, la narrativa sn generada y la consulta original q son introducidas como entrada del LLM empleando una segunda instrucción, $\text{promp}_{\text{alt}}$, que pide generar consultas alternativas teniendo en cuenta el contexto adicional proporcionado por sn:

$$q'_1, \ldots, q'_n = \text{LLM}(\text{promp}_{\text{alt}}(q, sn)). \tag{3.16}$$

Las narrativas generadas tienen como fin detallar y aclarar la necesidad de información expresada por el/la usuario/a, imitando sus contrapartes originales. Nuestra intención es obtener un texto similar a, por ejemplo, la narrativa que TREC HM 2022 da para la consulta "*hydroquinone banned europe*", que es "*Hydroquinone is used as a topical application in skin whitening to reduce the color of skin. This question is asking if European governments have banned the use of hydroquinone*".

Esta estructura en dos etapas, representada en la Figura 3.5 permite sustituir con facilidad la narrativa sintética sn con una narrativa real (esto es, escrita por

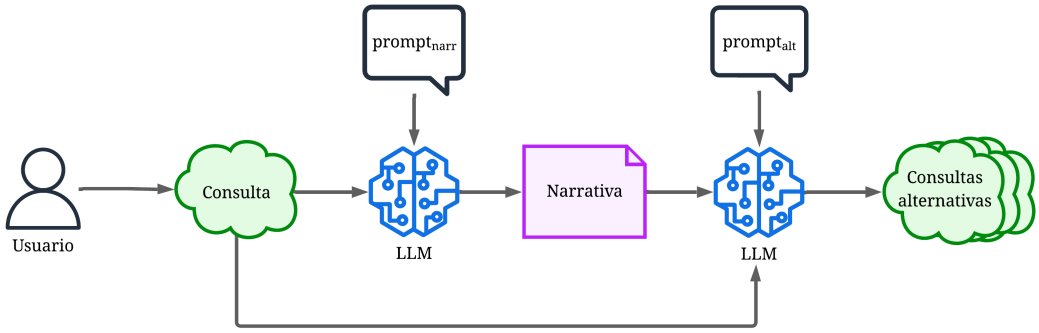


Figura 3.5: Esquema de etapas para la generación de consultas alternativas.

un humano específicamente para la consulta en consideración). Como consecuencia, podremos comparar la calidad de los resultados al usar unas u otras.

Otra de las ventajas de este método es que puede ser incorporado a un sistema de recuperación de información preexistente como un módulo independiente. Puesto que puede ser considerado un paso de preprocesamiento anterior a la ejecución de la búsqueda y/o el re-ranking llevados a cabo por el propio sistema, no es necesario realizar adaptaciones sobre estos procesos para beneficiarse de la mejora que puede aportar sobre su rendimiento base.

3.4.1. Generación de Narrativas Sintéticas

Para crear narrativas a partir de las consultas, hemos diseñado varias posibilidades para $\text{promp}_{\text{narr}}$, empleando estilos diferentes. A continuación, describimos cada uno de ellos, si bien solamente incluimos la instrucción explícita de aquel que fue seleccionado para el método final (estilo (b)), por la calidad superior de los resultados obtenidos (para más detalles al respecto, véase la Sección 4.3; el resto de instrucciones se incluyen en el apéndice C).

(a) Una instrucción concisa que pide generar una narrativa en un único párrafo que "detalle la necesidad de información" y "describa las características de documentos *helpful* y *harmful*".

(b) Una instrucción análoga a la de (a), pero que además solicita que la narrativa siga el formato empleado en TREC. Denotando por [consulta] el fragmento a sustituir por la consulta de entrada, el texto completo es:

> *Given the query* [consulta]*, write a narrative detailing the information need and describing the characteristics of helpful and harmful documents using the standard TREC format for narratives. Write one paragraph and do not repeat the query in your answer.*

(c) Una instrucción más detallada que pide que, en un solo párrafo, se detalle la necesidad de información, se emplee el formato de TREC y que se sigan una serie de indicaciones sobre la estructura, la voz, el tono y el estilo del lenguaje con el objetivo de obtener un texto neutral, informativo, objetivo y directo.

3.4.2. Generación de Consultas

Inspirados por Thomas et. al [60], que emplearon plantillas con diferentes configuraciones para la generación de *qrels* mediante LLMs, hemos diseñado nuestra propia plantilla para la instrucción $\text{promp}_{\text{alt}}$. Esta, que se muestra en la Figura 3.6, tiene una configuración basada en:

1. R, un parámetro binario que marca la presencia de un rol del sistema.
2. N, un parámetro binario que indica la presencia de una narrativa que expande y aclara la consulta de entrada. Nótese que, en caso de que sí se emplee una narrativa, esta puede o bien ser de procedencia sintética o bien ser la dada en los conjuntos de datos originales.
3. C, una variable que representa el uso de *chain-of-thought*. Si $C = 0$, no se incluye texto adicional y no se aplica la técnica. Si $C = 1$, se añade una instrucción genérica que solicita razonamiento progresivo y comprensión semántica [57]. Si $C = 2$, se amplía con detalles específicos: se pide al LLM que reflexione sobre el potencial de recuperación de documentos relevantes (útiles), correctos y creíbles de la consulta y que tenga en cuenta la importancia relativa de cada uno de estos aspectos.

Instruimos al LLM para que produzca n alternativas y que las devuelva como un *array* de *strings*.

Rol (R)	You are a search engineer trying to improve the relevance, correctness and credibility of search results for health-related queries.
	Given a query, you must provide a list of n alternative queries that express the same information need as the original one, but that are phrased in such a way that they are more likely to retrieve relevant, correct and credible documents.
Narrativa (N)	**Query:** A person has typed [*consulta*] into a search engine.
	They were looking for: [*narrativa*]
Chain of thought (C)*	**Instructions:** Let's think step by step: Consider the underlying intent of the search.
	Measure how prone the original query is to retrieve useful documents. Measure how prone the original query is to retrieve supportive documents for the correct treatment of the query's question.
	Measure how prone the original query is to retrieve credible documents.
	Consider the aspects above and the relative importance of each, and produce an array of variant queries without providing any reasoning. Example: ["query variant 1", "query variant 2", ...]

*La instrucción de *chain-of-thought* correspondiente a $C = 1$ es el texto subrayado en verde.
La correspondiente a $C = 2$ incluye el texto verde y el azul.

Figura 3.6: Plantilla empleada para generar consultas alternativas ($\text{prompt}_{\text{altq}}$). El texto en *cursiva* se sustituye por valores reales en las ejecuciones. Los parámetros R, N y C determinan si los fragmentos en color correspondientes se incluyen.

3.5. Predicción de Desinformación en Consultas

En esta sección describimos los métodos evaluados en cuanto a su capacidad de predicción de la presencia de desinformación en los resultados de búsquedas de consultas. Realizamos una distinción entre los métodos empleados como referencia, seleccionados con el objetivo de establecer la eficacia de estrategias de uso general en la QPP, y los métodos que proponemos específicamente para esta tarea, todos ellos basados en el uso de LLMs.

3.5.1. Métodos de Referencia

Con el objetivo de establecer mediciones de referencia, se han seleccionado una serie de métodos clásicos y ampliamente usados en la literatura de QPP, así como un método basado en un modelo neuronal del estado del arte en cuanto a análisis de consultas. Remarcamos que, dado que no existen publicaciones precedentes que analicen la capacidad de estos métodos para la estimación de la desinformación en contextos sanitarios, establecer estas referencias constituye un resultado novedoso en sí mismo.

Puesto que el método basado en LLMs que proponemos (descrito en la Subsección 3.5.2) es de naturaleza *pre-retrieval* (es decir, tiene como entrada la consulta para la cual se quiere realizar la estimación, pero no los resultados de búsqueda de la misma, en contraposición a las técnicas *post-retrieval*, que requieren conocer ambos), las referencias se han elegido dentro de esta categoría. El motivo para ello es que, por un lado, los métodos *post-retrieval* cuentan con más información para realizar las predicciones, lo que generalmente les da una mayor precisión y dificulta la comparación directa con técnicas *pre-retrieval*. Por otro lado, estas últimas tienen un mayor potencial de aplicación a casos de uso reales. Por ejemplo, podrían emplearse para decidir si una consulta merece ser reformulada o no en base a si sus resultados de búsqueda se estiman como significativamente dañinos o no. Aunque las técnicas *post-retrieval* también pueden usarse a tal efecto, el hecho de que requieran ejecutar la búsqueda con anterioridad las hace mucho más costosas computacionalmente.

Denotando por C el corpus empleado; por N, el número de documentos en C; por q, una consulta dada y que está compuesta de una serie de términos; y por $\mathrm{f}(t, C)$, el número de veces que aparece el término t en C, enumeramos y describimos a continuación los métodos de referencia clásicos seleccionados, siguiendo como referencia un artículo de Khodabakhsh et al. [63]:

(I) *Average Inverse Document Frequency* (avg IDF), la media de la puntuación

de IDF para cada término $t \in q$

$$\text{avg IDF} = \frac{1}{|q|} \sum_{t \in q} \text{IDF}(t), \tag{3.17}$$

donde recordamos que la fórmula de IDF viene dada por la ecuación (3.14).

(II) *Maximum Inverse Document Frequency* (max IDF), el máximo de la puntuación de IDF de los términos $t \in q$:

$$\text{max IDF} = \max_{t \in q} \text{IDF}(t). \tag{3.18}$$

(III) *Average Collection Query Similarity* (avg SCQ), la media de la puntuación de SCQ para cada término $t \in q$, donde SCQ se calcula como

$$\text{SCQ}(t) = (1 + \log(\text{f}(t, C))) \, \text{IDF}(t), \tag{3.19}$$

por lo que

$$\text{avg SCQ} = \frac{1}{|q|} \sum_{t \in q} (1 + \log(\text{f}(t, C))) \, \text{IDF}(t). \tag{3.20}$$

(IV) *Maximum Collection Query Similarity* (max SCQ), el máximo de la puntuación de SCQ de los términos $t \in q$:

$$\text{max SCQ} = \max_{t \in q} (1 + \log(\text{f}(t, C)) \, \text{IDF}(t). \tag{3.21}$$

(V) *Average Inverse Collection Term Frequency* (avg ICTF), que está dado por:

$$\text{avg ICTF}(q) = \frac{1}{|q|} \sum_{t \in q} \log \frac{N}{\text{f}(t, C)}. \tag{3.22}$$

(VI) *Simplified Clarity Score* (SCS), que se puede calcular como

$$\text{SCS}(q) = \log \frac{1}{|q|} + \text{avg ICTF}(q). \tag{3.23}$$

A la lista de predictores previamente mencionados incorporamos un método más sofisticado basado en un reciente modelo neuronal: el clasificador *Query-Quality-Classifier*[7], disponible en Hugging Face y desarrollado a partir del trabajo de Google AI Language [97]. Este modelo ha sido diseñado y entrenado específicamente para distinguir entre consultas bien y mal formuladas. Partiendo del supuesto de que una consulta mal formulada podría tener una presencia destacada de documentos *harmful* en sus resultados de búsqueda, incluimos como última métrica de referencia la confianza estimada por el clasificador de que la consulta introducida por el/la usuario/a esté bien formulada.

[7] https://huggingface.co/dejanseo/Query-Quality-Classifier

3.5.2. Métodos Propuestos con LLMs

Con el fin de explorar si los LLMs tienen la capacidad de predecir desinformación, hemos diseñado una instrucción que les solicita evaluar la calidad de consultas en un contexto sanitario. Partiendo de la hipótesis de que existe una relación entre el nivel de controversia de una consultas y la *harmfulness* de sus resultados de búsqueda, pedimos al LLM que estime, solo a partir del texto de la consulta (sin acceder a los resultados de búsqueda), el grado de controversia y otros factores parciales ligados al mismo.

Se probaron distintas formulaciones de la instrucción, como pedir una única puntuación o simular valoraciones de jueces independientes. Finalmente, se optó por promediar los resultados de varias ejecuciones con diferentes hiperparámetros, al ofrecer mejores resultados en las primeras iteraciones de diseño.

Tras fijar la instrucción de controversia, se creó una segunda variante que solicita, además de la puntuación global, un desglose de los factores parciales mencionados, que son ambigüedad, polarización en los resultados, potencial de desinformación e información contradictoria antes de dar el resultado final. Esta aproximación, similar a emplear *chain-of-thought prompting*, fuerza al modelo a evaluar cada aspecto y permite analizar su valor predictivo individual.

En ambas instrucciones, se requiere que las puntuaciones (globales y parciales) sean enteros en el intervalo $[1, 5]$, donde 1 correspondería a la evaluación más baja y 5, a la más alta. En la primera variante se devuelve un único valor por ejecución y en la segunda, un *array* de cinco enteros, con cuatro puntuaciones parciales y una total. La plantilla resultante se muestra en la Figura 3.7.

	You are an expert in information retrieval and search engine bias. Given a query, you must determine its level of controversy within the context of health-related information retrieval. Consider factors such as ambiguity, polarization in search results, potential misinformation and conflicting information. Think step by step and provide a score on an integer scale of 1 (not controversial) to 5 (highly controversial) for the query [*consulta_palabras_clave*].
Solo puntuación global	Your answer should be a single integer representing the total score. Do not include any other information.
Con puntuaciones parciales	Your answer should be a JSON array of scores for the individual factors and the total score at the end. Do not include any textual description. Example: [1, 5, 2, 2, 3]

Figura 3.7: Plantilla empleada para evaluar lo controversial que es una consulta. En cada ejecución, se incluyó solo uno de los dos fragmentos subrayados (el naranja para pedir únicamente una puntuación global, o el azul para pedir tanto una puntuación global como puntuaciones parciales).

Capítulo 4

Pruebas

Este capítulo se centra en describir el plan de pruebas que valida que se ha dado alcance a los objetivos planteados. Para ello, se presentará tanto el diseño y enfoque metodológico adoptado en los experimentos como los resultados obtenidos en su ejecución.

Este trabajo cubre tres estudios bien diferenciados, uno referente a la comparación de sistemas de búsqueda, otro a la generación de consultas alternativas y otro a la QPP enfocada a la desinformación, y los analizaremos por separado en las Secciones 4.2, 4.3 y 4.4, respectivamente. No obstante, debido a la presencia de paralelismos entre los tres experimentos y las dependencias entre los mismos, incluimos en primer lugar un apartado dedicado a describir consideraciones de tipo general (Sección 4.1).

4.1. Consideraciones Generales

Remarcamos primero una configuración fundamental que se mantendrá constante a lo largo de los experimentos: imitando el tipo de consultas que los/las usuarios/as introducen en buscadores web, emplearemos únicamente las consultas de los conjuntos de datos que estén en un **formato de palabras clave**. Nótese que este tipo de consultas habitualmente presenta una sintaxis formalmente incorrecta, al eludir aquellos términos que pueden ser inferidos por el contexto y al haber una ausencia marcada de conectores. Esta característica, junto a su brevedad, puede dificultar la interpretación de la necesidad de información formulada por el/la usuario/a y acrecentar problemas habituales en la RI, como el *lexical gap*. En la práctica, hemos comprobado que en la amplia mayoría de los

casos el rendimiento de los sistemas es más bajo para este tipo de consultas que cuando se emplea un formato de pregunta completa. Motivados por el potencial que puede tener mejorar sistemas de búsqueda en contextos de aplicación reales, hemos descartado el uso de consultas en formato de pregunta completa.

Por otro lado, las conclusiones extraídas de las pruebas relativas a la comparación de sistemas de búsqueda del estado del arte con respecto a la desinformación tienen un impacto significativo en los otros dos experimentos. Concretamente, querremos seleccionar un sistema con un desempeño representativo del estado del arte y que no tenga un tiempo de ejecución desorbitado. Por consiguiente, solamente indicaremos sus características técnicas en la Sección 4.2.

Destacamos también que **cada una de las posteriores secciones de este capítulo cuenta con su propio conjunto de métricas**, pues tienen tanto diferentes objetivos como marcos teóricos y experimentales. Para evaluar sistemas de búsqueda, tendremos en cuenta tanto las métricas de RI (previamente descritas en la Subsección 3.2.1) como las métricas de recuperación de desinformación (Subsección 3.2.2), con el fin de encontrar un modelo que sobresalga en ambos aspectos (y, de forma secundaria, tendremos en cuenta los tiempos de ejecución). Para evaluar la generación de consultas alternativas, utilizaremos únicamente las métricas de recuperación de desinformación, pues el foco de la técnica es generar consultas que recuperen menos información dañina. Para evaluar la predicción de desinformación en resultados de búsqueda, tendremos en cuenta las métricas de QPP diseñadas a este respecto y descritas en la Subsección 3.2.3.

De igual modo, **las colecciones de prueba también varían entre experimentos**. En las Secciones 4.2 y 4.3 hacemos uso de TREC HM 2020, TREC HM 2021, TREC HM 2022 y CLEF_v1. En contraposición, para las pruebas de la Sección 4.4 descartamos TREC HM 2020, ya que, al estar todas sus consultas centradas alrededor de un mismo tema (la COVID-19), estas tenían una proporción demasiado elevada de palabras en común. Esto impide valorar los resultados de los métodos de QPP de forma equivalente al resto de colecciones de prueba. Asimismo, para este experimento se emplea CLEF_v2, en lugar de CLEF_v1, buscando homogeneizar las colecciones lo máximo posible para evitar la influencia de condiciones externas.

4.2. Evaluación de Sistemas de Búsqueda

Para ejecutar y evaluar sistemas de búsqueda del estado del arte, se empleó la librería BEIR (*BEnchmarking Information Retrieval*) [74], una plataforma de Python estandarizada para este preciso fin. Aunque BEIR incluye 19 conjuntos

de datos, ninguno de estos está orientado al ámbito de la salud, por lo que hemos cargado manualmente las colecciones TREC HM 2020, TREC HM 2021, TREC HM 2022 y CLEF_v1.

Los modelos evaluados corresponden a los descritos en la Sección 3.3. Estos fueron probados con las versiones e hiperparámetros del Cuadro C.1 del apéndice C en modo *zero-shot*, puesto que ninguno de nuestros conjuntos de datos se utilizó para el entrenamiento de los modelos. Por otro lado, ejecutamos BM25 a través de su implementación de Pyserini, con parámetros $k_1 = 0.9$ y $b = 0.4$. Recordamos que, como indicamos en la Sección 3.3, el top 1000 recuperado por BM25 es la base sobre la que se ejecutan los modelos del Cuadro C.1, con la salvedad de que los modelos de re-ranking (ELECTRA, MiniLM, TinyBERT y MonoT5) solo operan sobre el top 100. DocT5query sigue una ligera variante de este esquema, pues al tener como foco la expansión de documentos, tomamos como sus resultados el ranking recuperado por BM25 sobre el corpus expandido.

4.2.1. Análisis de los Modelos de Búsqueda

Como se puede ver en los cuadros 4.1 y 4.2, los modelos de re-ranking tienen un rendimiento superior que los dispersos, densos y que BM25, tanto con respecto a las métricas de RI (P@K, Recall@K, AP@K y NDCG@K, donde hemos fijado los umbrales $K = 10$, $K = 100$ y $K = 1000$) como en *compatibility*. Entre ellos, es difícil establecer un claro favorito, pues su desempeño relativo no es

Cuadro 4.1: Resultados de la evaluación de las métricas de RI más destacadas sobre los modelos considerados. Por razones de espacio, reportamos únicamente los datos obtenidos para TREC HM 2022. Cada valor se obtiene como el promedio de las evaluaciones obtenidas sobre las 50 consultas de la colección.

Tipo de sistema de recuperación	**Sistema de recuperación**	**P**@10	**Recall**@10	**MAP**@100	**MAP**@1000	**NDCG**@100	**NDCG**@1000
Léxico	BM25	.567	.042	.136	.235	.355	.235
Disperso	SPARTA	.456	.034	.087	.157	.290	.157
	SPLADE	.460	.034	.084	.150	.280	.436
	DocT5query	.558	.043	.128	.223	.337	.533
Denso	DPR	.238	.017	.032	.059	.165	.266
	ANCE	.540	.042	.111	.173	.330	.448
	TAS-B	.493	.037	.098	.168	.310	.457
Re-ranking	ELECTRA-base	.622	.048	.143	.242	.372	.555
	MiniLM-L-4-v2	.551	.043	.136	.235	.364	.548
	MiniLM-L-6-v2	.560	.045	.138	.237	.371	.555
	MiniLM-L-12-v2	.560	.044	.135	.234	.362	.468
	TinyBERT-L-2-v2	.562	.043	.135	.234	.365	.550
	TinyBERT-L-4	.531	.041	.130	.229	.354	.540
	TinyBERT-L-6	.562	.042	.133	.232	.360	.545
	MonoT5 (base)	.591	.045	.137	.236	.368	.551
	MonoT5 (large)	.618	.049	.143	.242	.373	.555
	MonoT5 (base-med)	.593	.046	.138	.237	.367	.551

Cuadro 4.2: Resultados de *compatibility* de los modelos considerados. Para *compatibility helpful* y *harmful* se da el promedio sobre las consultas de cada colección.

Tipo de sistema de recuperación	Sistema de recuperación	TREC HM 2020			TREC HM 2021		
		Help	Harm	Help-Harm	Help	Harm	Help-Harm
Léxico	BM25	.214	.047	.167	.129	.145	-.016
Disperso	SPARTA	.212	.092	.120	.100	.082	.018
	SPLADE	.224	.092	-.696	.083	.068	.015
	DocT5query	.185	.041	.144	.173	.266	-.093
Denso	DPR	.119	.024	.095	.050	.046	.004
	ANCE	.142	.058	.084	.099	.078	.021
	TAS-B	.060	.011	.049	.095	.081	.014
Re-ranking	ELECTRA-base	.203	.078	.125	.145	.138	.007
	MiniLM-L-4-v2	.237	.089	.148	.132	.133	-.001
	MiniLM-L-6-v2	.211	.082	.129	.129	.145	-.016
	MiniLM-L-12-v2	.226	.078	.148	.132	.136	-.004
	TinyBERT-L-2-v2	.244	.088	.156	.121	.131	-.010
	TinyBERT-L-4	.251	.094	.157	.119	.125	-.006
	TinyBERT-L-6	.235	.086	.149	.129	.133	-.004
	MonoT5 (base)	.260	.103	.157	.126	.133	-.007
	MonoT5 (large)	.263	.107	.156	.132	.132	0
	MonoT5 (base-med)	.265	.109	.156	.131	.134	-.003

Tipo de sistema de recuperación	Sistema de recuperación	TREC HM 2022			CLEF_v1		
		Help	Harm	Help-Harm	Help	Harm	Help-Harm
Léxico	BM25	.173	.144	.029	.101	.272	-.171
Disperso	SPARTA	.147	.115	.032	.041	.106	-.065
	SPLADE	.155	.096	.059	.057	.137	-.080
	docT5query	.155	.154	.001	.091	.246	-.155
Denso	DPR	.075	.049	.026	.028	.062	-.034
	ANCE	.178	.120	.058	.045	.127	-.082
	TAS-B	.154	.091	.063	.049	.107	-.058
Re-ranking	electra-base	.201	.125	.076	.100	.202	-.102
	MiniLM-L-4-v2	.184	.136	.048	.094	.208	-.114
	MiniLM-L-6-v2	.195	.126	.069	.094	.212	-.118
	MiniLM-L-12-v2	.179	.131	.048	.095	.211	-.116
	TinyBERT-L-2-v2	.179	.125	.054	.086	.211	-.125
	TinyBERT-L-4	.171	.116	.055	.080	.212	-.132
	TinyBERT-L-6	.196	.109	.087	.083	.213	-.130
	MonoT5 (base)	.193	.134	.059	.099	.208	-.109
	MonoT5 (large)	.193	.138	.055	.096	.209	-.113
	MonoT5 (base-med)	.192	.136	.056	.099	.208	-.109

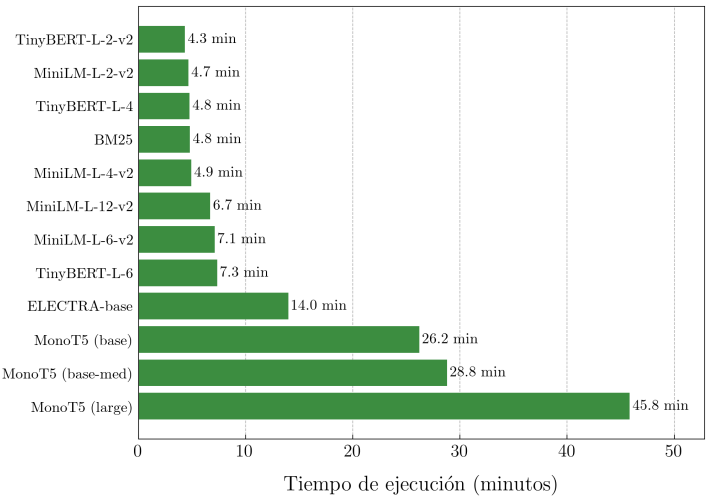


Figura 4.1: Comparación del tiempo de ejecución (sobre las consultas de la colección TREC HM 2020) de BM25 y de los modelos de re-ranking seleccionados.

consistente entre las colecciones de prueba fijadas. Teniendo en cuenta este hecho, así como el tiempo de ejecución de cada sistema (que nos será relevante por el elevado número de casos de prueba en experimentos posteriores, y que aparece representado en la Figura 4.1), hemos optado por seleccionar MiniLM-L-12-v2 como el modelo representativo del estado del arte en la RI enfocada a la salud. Asimismo, consideramos sorprendentes los resultados que ha obtenido BM25, que a pesar de su sencillez logra igualar o incluso superar a varios modelos neuronales dispersos y densos, mucho más costosos computacionales. Por estos motivos, lo utilizaremos también como modelo de referencia en experimentos posteriores.

4.3. Generación de Consultas Alternativas

Como modelo para la generación de narrativas y consultas, optamos, en primer lugar, por el LLM GPT-4, de OpenAI. Su hiperparámetro de temperatura (un valor entre 0 y 2 que controla la aleatoriedad de la salida del modelo, de tal forma que esta es más aleatoria y "creativa" si la temperatura se acerca a 2, y más determinista y "predecible" si se acerca a 0) fue fijado a un valor bajo, de 0.2, con el objetivo de fomentar que el texto generado fuera preciso y que se ciñera a elecciones de tokens probables. Asimismo, la penalización de frecuencia (un valor entre -2 y 2 que, cuando es positivo, disminuye la probabilidad de que se repitan tokens y, cuando es negativo, aumenta dicha probabilidad) se fijó a 0, con el objetivo de que las repeticiones de tokens no fueran un factor relevante en el proceso de generación. Nos mantenemos agnósticos con respecto al resto de los hiperparámetros, que se dejaron en sus valores por defecto.

Como se explicó en la Subsección 3.4.2, la generación de consultas depende de una configuración compuesta de tres parámetros: R (rol), N (narrativa, que, cuando está presente, puede ser real o sintética) y C (*chain-of-thought*). R y N son parámetros binarios, mientras que C puede tomar los valores 0, 1 y 2.

Para cada una de las posibles configuraciones de parámetros (en los casos en los que fueran posibles, ya que la colección CLEF IR 2016 no disponía de narrativas reales) se generaron cinco consultas alternativas. Es decir, en la plantilla de la Figura 3.6, el parámetro n vale siempre 5. La motivación detrás de esta decisión es que, debido a la naturaleza no determinística de los LLMs, es importante aumentar el tamaño muestral para reforzar la confianza en las métricas calculadas. Además, al ser el total de configuraciones y consultas bastante considerable (18 configuraciones por cada sistema de búsqueda y 450 consultas entre los cuatro conjuntos de datos) y tras comprobar cualitativamente que no es necesario elevar el número de consultas alternativas para obtener una muestra representativa de interpretaciones y variantes de una consulta dada, se ha descartado aumentar el

valor de n.

Si la salida del LLM tenía errores en el formato de salida, p.e., no era el propio de una lista de cinco elementos de carácter textual, repetimos el envío de la instrucción, esta vez con una línea adicional que enfatiza el uso de dicho formato. Este tipo de errores fueron suficientemente escasos como para optar por no tomar medidas adicionales.

Para generar las narrativas sintéticas también se empleó GPT-4. Se probaron las diferentes instrucciones diseñadas a este respecto y descritas en la Subsección 3.4.1, pero la que dio lugar a las consultas con mejores resultados de *compatibility* empleando diferentes configuraciones de R, N y C fue la variante que pide que se emplee el formato de narrativas de TREC y no da pautas explícitas de estilo, sino que se limita a indicar que se debe "detallar la necesidad de información" y "describir las características de documentos *helpful* y *harmful*". En base a esto, se fijó dicha instrucción para la generación de narrativas sintéticas, y esta ha sido la empleada para reportar los resultados finales de nuestro método.

Como referencia, se empleó la *compatibility* del top 1000 de BM25 y MiniLM-L-12-v2. Hemos comprobado además si la mejora sobre estas referencias, empleando el mismo sistema de búsqueda, es estadísticamente significativa. Para ello, se ha utilizado un test de Wilcoxon unilateral con nivel de significancia $\alpha = .05$.

4.3.1. Resultados

El Cuadro 4.3 presenta los resultados obtenidos, limitados a la configuración $C = 1$ por razones de espacio, aunque las diferencias entre opciones de *chain-of-thought* fueron despreciables. Notablemente, hay mejoras significativas al realizar búsquedas con las consultas alternativas, tanto con BM25 como con MiniLM-L-12-v2. Aunque las narrativas reales producen mejores resultados, las sintéticas ofrecen un rendimiento comparable y satisfactorio. Obsérvese, además, que la puntuación agregada de *compatibility helpful-harmful* aumenta para casi todos los casos de prueba con consultas alternativas cuando se usa MiniLM-L-12-v2. Con BM25 la tendencia se mantiene, pero esta es más atenuada, ya que, aunque reduce los *harmful*, lo hace a costa de una caída aún mayor en los *helpful*, principalmente debido a su potencia menor con respecto a MiniLM-L-12-v2. La inclusión de un rol (R) tiene un impacto marginal y poco concluyente (lo cual es consistente con los hallazgos extraídos del estudio de Thomas et al. [60], que reportaban efectos modestos y dependientes del contexto de *prompting*).

Por consiguiente, concluimos que la generación de consultas alternativas con la adición de narrativas resulta beneficiosa. En la Figura 4.2 se puede apreciar

Cuadro 4.3: *Compatibility helpful*, *harmful* y *helpful-harmful* para cada configuración posible de *prompting* en el proceso de generación de consultas (parámetros R, N y C y uso de narrativa real o sintética), empleando GPT-4. Cada valor de *compatibility helpful* y *harmful* reportado es el promedio de los valores de cinco ejecuciones con la configuración y sistema de búsqueda correspondientes (salvo en el caso de las consultas originales, donde el proceso es determinístico). Para cada bloque, conjunto de datos y métrica, se destaca el mejor resultado en negrita. Las puntuaciones de *compatibility helpful* y *harmful* se marcan con * cuando la mejora sobre la referencia es estadísticamente significativa, empleando el test de Wilcoxon unilateral con p-valor $< .05$.

Modelo	Consultas			TREC HM 2020			TREC HM 2021		
	R	**N**	**C**	Help	Harm	Help - Harm	Help	Harm	Help - Harm
BM25	originales			.214	.047	.167	**.129**	.145	-.016
sin narrativa	0	0	1	.236	.058	.178	.109	.108	.001
	1	0	1	.232	.058	.174	.105	.112	-.007
narrativa real	0	1	1	.256	.052	.204	.103	**.098***	.005
	1	1	1	**.259**	.048	**.211**	.118	.100*	**.018**
narrativa sintética	0	1	1	.191	.046	.145	.098	.102	-.004
	1	1	1	.197	**.045**	.152	.106	.103	.003
MiniLM-12	originales			.226	**.078**	.148	.132	.136	-.004
sin narrativa	0	0	1	.289*	.089	.200	.135	**.129**	.006
	1	0	1	.288*	.092	.196	.137	.134	.003
narrativa real	0	1	1	.307*	.092	.215	.142*	.134	**.008**
	1	1	1	**.312***	.089	**.223**	**.144**	.138	.006
narrativa sintética	0	1	1	.274*	.087	.187	.136	.132	.004
	1	1	1	.274*	.085	.189	.138	.134	.004

Modelo	Consultas			TREC HM 2022			CLEF_v1		
	R	**N**	**C**	Help	Harm	Help - Harm	Help	Harm	Help - Harm
BM25	originales			**.173**	.144	.029	**.101**	.272	-.172
sin narrativa	0	0	1	.100	.077*	.023	.086	.179*	-.093
	1	0	1	.112	.077*	.035	.085	.182*	-.097
narrativa real	0	1	1	.086	.080*	.006	——	——	——
	1	1	1	.100	.085*	.015	——	——	——
narrativa sintética	0	1	1	.072	**.049***	.023	.089	**.155***	**-.066**
	1	1	1	.095	.056*	**.039**	.087	.157*	-.070
MiniLM-12	originales			.179	**.131**	.048	.095	.211	-.116
sin narrativa	0	0	1	.190	.140	.050	.096	.198*	-.102
	1	0	1	.191*	.138	.053	.096	.200*	-.104
narrativa real	0	1	1	**.199***	.137	**.062**	——	——	——
	1	1	1	.195*	.144	.051	——	——	——
narrativa sintética	0	1	1	.188	.134	.054	**.100***	**.196**	**-.096**
	1	1	1	.189*	.135	.054	.099*	.196*	-.097

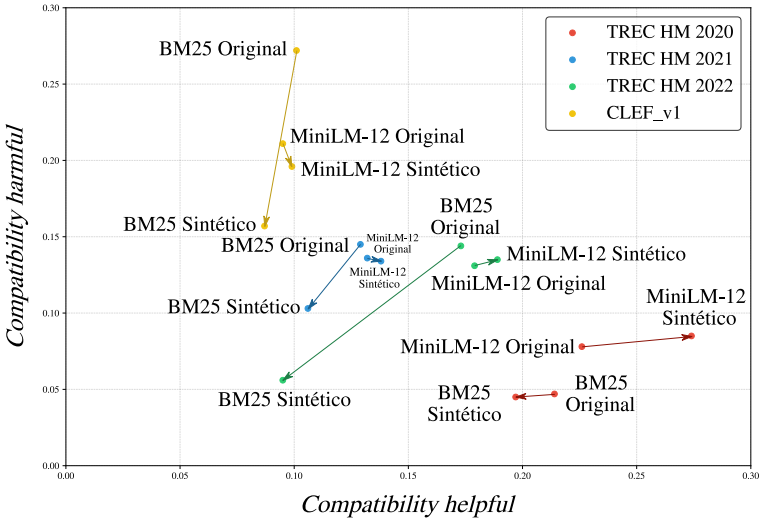


Figura 4.2: Comparación de los resultados de consultas alternativas generadas con narrativas sintéticas sobre las consultas originales, con configuración de parámetros $R = 1$, $N = 1$ y $C = 1$. El desempeño ideal corresponde a la esquina inferior-derecha, donde *compatibility helpful* $= 1$ y *compatibility harmful* $= 0$.

un análisis detallado de la diferencia entre usar consultas alternativas con narrativas sintéticas (específicamente, con $R = 1$, $N = 1$, $C = 1$) o las consultas originales sobre los cuatro conjuntos de datos y con los dos sistemas de búsqueda seleccionados. Se utilizan flechas para remarcar el cambio del desempeño original al obtenido con nuestra técnica. Es claro que BM25 disminuye la *compatibility harmful* en todos los casos, si bien también reduce la *compatibility helpful*. No obstante, esta reducción es menos pronunciada, de donde inferimos que el impacto global es positivo. En el caso de MiniLM-L-12-v2, hay un incremento de la *compatibility helpful* consistente en las cuatro colecciones y, adicionalmente, en dos de ellas (TREC HM 2022 y CLEF_v1) reduce la *compatibility harmful*. Por tanto, también en este caso podemos valorar positivamente el efecto global de la técnica, si bien existe un margen de mejora que puede ser interesante explorar en trabajos futuros, esto es, buscar causas y soluciones para los casos en los que se recuperan menos documentos *helpful* y/o más documentos *harmful*.

Tras los experimentos con GPT-4, se repitieron los mismos pasos con LLaMA3 para validar nuestras conclusiones también con un modelo de código abierto. Los valores de *compatibility helpful* y *compatibility harmful* son menores, que es razonable suponer que se debe a una menor calidad de las consultas generadas en términos de RI, por su relación con estas métricas. Por razones de espacio, los datos obtenidos se derivan al apéndice C.

4.3.2. Análisis Cualitativo

Hemos llevado a cabo un estudio de las consultas generadas por GPT-4 que daban lugar a las mayores mejoras en efectividad sobre sus contrapartes originales. Observamos que, frecuentemente, los incrementos destacados en *compatibility helpful* están asociados a haber clarificado la intención de búsqueda. Por ejemplo, "*Hib vaccine COVID-19*" (BM25: .015, MiniLM-L-12-v2: .084) tiene como alternativa "*Does the hib vaccine provide protection against COVID-19?*" (BM25: .894, MiniLM-L-12-v2: .536), y para "*Inhalers COVID-19*" (BM25: .038, MiniLM-L-12-v2: .154) se propone "*Effectiveness of inhalers for COVID-19 symptoms*" (BM25: .886, MiniLM-L-12-v2: .604).

Por otro lado, aquellas variantes que más disminuyeron la *compatibility harmful* con respecto a las consultas originales suelen añadir términos y expresiones que hacen énfasis en la seguridad y las pruebas científicas. Este efecto se puede notar en "*baking soda cancer*" (BM25: .492, MiniLM-L-12-v2: .614), que es transformada en "*baking soda cancer prevention evidence from health organizations*" (BM25: .04, MiniLM-L-12-v2: .416), o en "*Breast milk COVID-19*" (BM25: .069, MiniLM-L-12-v2: .229), para la que se propone "*Is it safe to breastfeed if I have COVID-19?*" (BM25: 0, MiniLM-L-12-v2: .03).

La intersección de ambos efectos, esto es, un aumento de *compatibility helpful-harmful*, se caracteriza por clarificar la intención de búsqueda y orientarla hacia fuentes acreditadas. Por ejemplo, "*magnetic wrist straps arthritis*" (BM25: −.179, MiniLM-L-12-v2: −.039) se transforma en "*scientific studies on magnetic wristbands for arthritis treatment*" (BM25: .241, MiniLM-L-12-v2: .494), y "*tylenol osteoarthritis*" (BM25: −.302, MiniLM-L-12-v2: −.35), en "*Tylenol dosage and side effects for osteoarthritis*" (BM25: .112, MiniLM-L-12-v2: −.049). No obstante, introducir este tipo de características no siempre conlleva una mejora en los resultados, como sucede en el caso de "*vitamin d asthma attacks*" (BM25: .563, MiniLM-L-12-v2: .703), reformulada como "*Scientific studies on vitamin D and asthma prevention*" (BM25: .026, MiniLM-L-12-v2: .506). Esto abre las puertas a futuros estudios que profundicen sobre las propiedades lingüísticas de consultas que favorecen la recuperación de contenido *helpful* y penalizan contenido *harmful*.

Existen también diferencias cualitativas entre las consultas de LLaMA3 y GPT-4. Generalmente, las consultas de GPT-4 tienen una sintaxis más completa (que favorece la RI) y un lenguaje más técnico y preciso. Por ejemplo, con $R = 1$, $N = 1$, narrativas sintéticas y $C = 1$, para "*bananas diabetes*" GPT-4 ofrece la alternativa "*How do bananas affect blood sugar levels in people with diabetes?*" y LLaMA3, "*bananas and blood sugar control*". Para "*Type O blood COVID-19*", GPT-4 propone "*Scientific studies on Type O blood and COVID-19 susceptibility*" y LLaMA3, "*COVID-19 and Type O blood symptoms*".

4.4. Predicción de Desinformación en Consultas

Los experimentos relativos a la predicción de desinformación en consultas incluían tanto la evaluación de métodos de QPP de referencia (seis métodos tradicionales en el campo y uno diseñado a partir de un modelo neuronal del estado del arte para el análisis de consultas, como se detalló en la Subsección 3.5.1) como el análisis de rendimiento de predictores que habíamos diseñado *ad hoc* para esta tarea. Para los segundos, se optó por emplear GPT-4 como LLM a través del cual realizar las generaciones pertinentes, ya que la dificultad de las tareas de QPP similares a esta (p.e., [68]) justifica la elección de un modelo lo más potente posible, como es el caso de GPT-4.

Asimismo, consideramos dos sistemas principales para predecir la desinformación en consultas con LLMs. En el primero, al que nos referiremos como **"Controversia"**, se utiliza una instrucción que pide evaluar la controversia de las consultas. En el segundo, siguiendo una idea análoga a la de las estrategias CoT, la instrucción pide que se analicen cuatro puntuaciones parciales (**ambigüedad**, **polarización**, **potencial de desinformación** e **información contradictoria**) antes de dar la evaluación de controversia final (a la que llamaremos **"Controversia CoT"**). Por lo tanto, considerando también las puntuaciones parciales, hay un total de seis métodos a analizar entre los dos sistemas.

Cada uno de estos dos tipos de instrucciones es ejecutado un total de cinco veces, en instancias del LLM independientes. Como en el caso del experimento de generación de consultas alternativas, argumentamos que es preferible contar con varias muestras y usar un resultado agregado, para mitigar los efectos de la aleatoriedad inherente a los LLMs (de hecho, esto fue comprobado experimentalmente antes de fijar el diseño final de nuestras técnicas), y que cinco muestras es suficiente a este respecto. Adicionalmente, para evitar generaciones demasiado homogéneas entre reevaluaciones de una misma consulta, las instancias del LLM tienen hiperparámetros de temperatura distintos: 0.2, 0.375, 0.55, 0.725 y 0.9, respectivamente. La penalización de frecuencia se mantiene en todas ellas a 0.

Por consiguiente, la correlación que reportamos para cada uno de nuestros métodos es la correlación de los promedios de los cinco valores generados para dicho método sobre cada consulta con respecto a los valores de *compatibility-harmful* reales de los resultados de un sistema de búsqueda fijo.

Para establecer referencias base, se han ejecutado seis métodos clásicos de QPP (Avg IDF, Max IDF, Avg SCQ, Max SCQ, SCS y Avg ICTF) y uno basado en el clasificador Query-Quality-Classifier. Dado que todos ellos son determinísticos, su valor para cada consulta solo se ha calculado una vez.

4.4.1. Resultados

En el Cuadro 4.4 mostramos los resultados obtenidos. En general, nuestros métodos obtienen correlaciones más robustas, alcanzando valores de hasta .583 (concretamente, para ambigüedad, BM25, TREC HM 2022 y el coeficiente ρ de Spearman), nada desdeñables en el contexto de tareas QPP [61], y especialmente en la categoría de QPP *pre-retrieval* [63, 98, 67]. En comparación, los métodos tradicionales muestran resultados más modestos, pero no alejados de su rendimiento habitual a la hora de predecir métricas de RI en tareas de QPP, por lo que ciertamente se pueden considerar prometedores también en la predicción de desinformación asociada a consultas. El clasificador probado, Query-Quality-Classifier, es el método con correlaciones más débiles. Estas son lo suficientemente bajas como para concluir que el modelo no generaliza a esta tarea. Aunque no podemos descartar que la corrección gramatical de una consulta no tenga influencia sobre la desinformación presente en los resultados de búsqueda de la misma, es posible que este sea un factor débil, que tenga un impacto más complejo del hipotetizado y/o que el modelo probado no sea lo suficientemente preciso al capturarlo para que se vea su efecto en esta tarea.

A excepción de Query-Quality-Classifier, todas las correlaciones obtenidas para BM25 tienen signo positivo. Esto también se cumple casi siempre para MiniLM-L-12-v2, salvo para los coeficientes τ de Kendall y ρ de Spearman de Max IDF, Max SCQ y SCS en TREC HM 2022, que no obstante son suficientemente bajos para ser desestimados y considerados casos aislados. El hecho de que las correlaciones de los nuestros métodos sean positivas es consistente con su propio diseño, pues tratan de localizar y evaluar aspectos que pueden afectar negativamente a la calidad de sus resultados. Que las correlaciones de los métodos clásicos sean positivas puede estar relacionado con su capacidad para predecir la RI, ya que, para que se considere que los documentos recuperados son *harmful*, estos deben ser relevantes. Esta explicación es consistente con que sus correlaciones no sean demasiado inferiores a las que suelen obtener en estudios de la literatura orientados a RI [67]. Por otro lado, que el signo de Query-Quality-Classifier fluctúe es poco indicativo, ya que la mayoría de sus resultados son demasiado cercanos a 0 como para extraer conclusiones a este respecto.

Puede observarse también que los métodos clásicos son peores a la hora de predecir MiniLM-L-12-v2 que BM25, que es consistente con el estudio de Faggioli et al. [67] en el que, en referencia a la QPP orientada a la RI, comprobaba que los modelos neuronales son más difíciles de predecir que los modelos de tipo "bolsa de palabras" (*bag-of-words*) como BM25. Nuestros métodos también muestran esta tendencia, salvo en el caso de TREC HM 2021, lo cual abre las puertas a analizar su potencial con respecto a modelos avanzados donde históricamente se ha obtenido un peor desempeño.

Otro caso peculiar es el de la colección de prueba CLEF_v2, la única en la que no solo los métodos basados en LLMs obtienen correlaciones estadísticamente significativas, y donde los métodos clásicos obtienen sus mejores resultados. Otro punto de interés es que es solo en esta colección donde Max IDF y Max SCQ obtienen mejores resultados que sus contrapartes, Avg IDF y Avg SCQ. Creemos que esto puede deberse a la mayor longitud de las consultas de la colección y al hecho de que la distribución de documentos *helpful* y *harmful* es diferente a la de TREC HM 2021 y TREC HM 2022, que son mucho más homogéneas entre sí, aunque sería necesario un estudio más profundo para confirmar que estas diferencias son determinantes.

Cuadro 4.4: Desempeño de los métodos de QPP seleccionados en términos de los coeficientes de correlación ρ de Pearson, τ de Kendall y ρ de Spearman entre la puntuación estimada para las consultas de tres conjuntos de datos y la *compatibility harmful* real de sus resultados de búsqueda. Distinguimos entre dos sistemas de búsqueda, BM25 y MiniLM-L-12-v2, y dividimos los métodos de QPP probados en dos grupos: los propuestos, basados en LLMs, y los de referencia. Para cada sistema de búsqueda, conjunto de datos y tipo de correlación, se destaca el mejor resultado en negrita (en caso de empate, los mejores resultados). * indica una correlación estadísticamente significativa con p-valor $< .05$.

Sistema de búsqueda	**Método QPP**	**TREC HM 2021**			**TREC HM 2022**			**CLEF_v2**		
		P-ρ	**K-τ**	**S-ρ**	**P-ρ**	**K-τ**	**S-ρ**	**P-ρ**	**K-τ**	**S-ρ**
BM25	Controversia	.292	.117	.127	.344*	.367*	.483*	.308*	.208*	.287*
	Controversia CoT	.331	.237	.307	.361*	.380*	.496*	.263*	.187*	.260*
	Ambigüedad	.269	.223	.290	**.445***	**.461***	**.583***	.265*	.201*	.278*
	Polarización	.345	.221	.300	.277	.292*	.393*	.292*	.209*	.288*
	Potencial de desinformación	.291	.233	.290	.382*	.370*	.499*	.270*	.187*	.266*
	Información contradictoria	**.369***	.198	.268	.409*	.394*	.538*	.270*	.178*	.251*
	Avg IDF	.156	**.237**	.322	.208	.114	.154	.174*	.168*	.246*
	Max IDF	.141	.161	.232	.191	.054	.087	.248*	.232*	.344*
	Avg SCQ	.201	.220	.325	.208	.133	.193	.140	.131*	.189*
	Max SCQ	.182	.173	.255	.172	.051	.072	**.369***	**.299***	**.417***
	SCS	.151	**.237**	**.336**	.103	.043	.050	.152	.124*	.181*
	Avg ICTF	.147	.216	.324	.166	.102	.126	.154	.147*	.216*
	Query-Quality-Classifier	-.016	.109	.166	-.217	-.124	-.158	-.059	-.038	-.049
MiniLM-L-12-v2	Controversia	.290	.143	.169	.215	.207	.317	.245*	.136*	.189*
	Controversia CoT	.309	.285*	.356*	.237	.214	.314	.198*	.127*	.176*
	Ambigüedad	.244	.271*	.343	**.312**	**.297***	**.420***	.211*	.153*	.213*
	Polarización	.325	.258	.324	.153	.153	.222	.234*	.140*	.193*
	Potencial de desinformación	.335	**.302***	**.363***	.274	.210	.315	.199*	.117*	.167*
	Información contradictoria	**.353***	.246	.308	.272	.205	.317	.198*	.103	.147
	Avg IDF	.104	.126	.200	.177	.056	.073	.136	.120*	.170*
	Max IDF	.083	.091	.125	.064	-.023	-.047	.214*	.176*	.269*
	Avg SCQ	.129	.117	.199	.192	.068	.111	.102	.092	.130
	Max SCQ	.108	.095	.140	.051	-.026	-.053	**.286***	**.229***	**.326***
	SCS	.133	.183	.254	.100	-.009	-.012	.138	.096	.136
	Avg ICTF	.122	.130	.212	.158	.049	.064	.117	.099	.140*
	Query-Quality-Classifier	.109	.179	.268	-.155	-.003	.014	-.050	-.036	-.065

Capítulo 5

Discusión de los Resultados

En este capítulo se hará una recapitulación de las conclusiones extraídas a partir de los resultados presentados en el capítulo 4, a la vez que se juzga el cumplimiento de las hipótesis propuestas en el capítulo 1.

De la comparativa entre sistemas de búsqueda en el ámbito de la salud concluimos que, como anticipábamos en la hipótesis (I), hay una clara correspondencia entre el desempeño de los modelos neuronales en tareas de RI y su capacidad para distinguir entre información *helpful* y *harmful*. En particular, hemos comprobado que los modelos de re-ranking superan ampliamente tanto a modelos clásicos (BM25) como a modelos del estado del arte dispersos y densos. En este sentido, reforzamos hallazgos de estudios previos sobre el rendimiento de sistemas de búsqueda para la RI, como los presentados por Thakur et al. [74], al examinar los sistemas sobre nuevas colecciones de prueba. Asimismo, este trabajo amplía dicho conocimiento al contexto específico de la salud, una comparativa que no había sido abordada de forma sistemática hasta ahora.

Por otro lado, los resultados obtenidos en los experimentos de generación de consultas alternativas con LLMs avalan la hipótesis (II). Así, confirmamos que los LLMs pueden ser incorporados a *pipelines* con modelos de búsqueda para favorecer la recuperación de documentos relevantes, correctos y creíbles y reducir la recuperación de documentos que contengan desinformación. Hemos validado esta hipótesis tanto para un modelo tradicional con una gran velocidad computacional como para uno de los modelos más avanzados del estado del arte y que, además, es especialmente destacado tanto en la RI como en distinguir información *helpful* de *harmful*, demostrando que nuestro método es efectivo tanto sobre sistemas de búsqueda simples como sobre sistemas avanzados de alto nivel técnico. En la línea de estudios como el de Bacciu et al. [55], reafirmamos que los LLMs son soluciones apropiadas para tareas de recomendación de consultas y

además, extendemos dicha afirmación a contextos específicos como el sanitario.

En cuanto a la hipótesis (III), corroboramos que el uso de narrativas reales repercute favorablemente sobre la generación de consultas. Disponer de este contexto adicional ayuda a los LLMs a clarificar la intención de búsqueda del/de la usuario/a, y esto se ve reflejado positivamente sobre su desempeño, que es consistente con lo encontrado por Thomas et al. en tareas de RI [60]. En ausencia de narrativas reales, observamos que estas pueden ser satisfactoriamente sustituidas por narrativas sintéticas, aunque su contribución es más débil. No obstante, han demostrado ser una estrategia de notable potencial.

Por otra parte, hemos demostrado que los métodos de QPP clásicos poseen una cierta capacidad para predecir la presencia de desinformación en los resultados de búsqueda de consultas. Como conjeturábamos en la hipótesis (IV), su efectividad parece estar muy ligada a la relación entre *harmfulness* y relevancia, de modo que, aunque validamos la hipótesis, concluimos que su capacidad predictiva frente a la desinformación es limitada. No obstante, nuestro análisis abre una nueva línea de investigación al aplicar la QPP al ámbito de la salud, y demuestra que los enfoques tradicionales pueden adaptarse a él con solo pérdidas moderadas de eficacia frente a estudios como los de Faggioli et al. [67], Khodabakhsh et al. [63] o Saleminezhad et al. [98].

Sin embargo, los métodos específicos a la predicción de *harmfulness* para consultas que proponemos, basados en la evaluación de la controversia potencial de las consultas a través de LLMs, obtienen resultados más robustos y, en numerosos casos, estadísticamente significativos. Esto confirma la hipótesis (V), reafirmando que los LLMs pueden anticipar eficazmente el riesgo de desinformación incluso antes de ejecutar las búsquedas, lo cual mejora significativamente el coste computacional frente a soluciones basadas en LLMs que sí requieren ejecutarlas, como la propuesta por Arabzadeh et al. [68].

Capítulo 6

Conclusiones y Posibles Ampliaciones

La desinformación representa un peligro social que, en la actualidad, se ve acrecentado por la disponibilidad de recursos y fuentes online por los que diariamente circula información falsa y/o inexacta. Su influencia sobre el comportamiento y la toma de decisiones de los/las usuarios/as pone en el punto de mira aquellos algoritmos que permiten la difusión de este contenido en ámbitos especialmente sensibles, como el de la salud. Por consiguiente, el estudio y el desarrollo de sistemas de búsqueda capaces de detectar y mitigar la presencia de desinformación en sus resultados es de suma importancia para la protección de los/las usuarios/as.

Este trabajo fue planteado con el objetivo de contribuir al progreso en este problema abierto, buscando mejorar el conocimiento de la comunidad científica sobre la desinformación y desarrollar técnicas que permitan evitar su difusión. Para ello, se ha llevado a cabo un estudio en tres frentes.

En primer lugar, se ha comprobado que los modelos de búsqueda del estado del arte tienen capacidad y potencial a la hora de favorecer información relevante, correcta y creíble (*helpful*) sobre aquella relevante, pero incorrecta (*harmful*). Se ha observado que los modelos de re-ranking son especialmente eficaces a este respecto, frente a arquitecturas como las de sistemas dispersos y densos, más ligeros computacionalmente, pero menos competentes en esta tarea.

En segundo lugar, se ha diseñado una técnica basada en LLMs para transformar las consultas de usuarios/as en variantes menos propensas a recuperar desinformación. Bajo la suposición de que sería positivo guiar su generación con explicaciones detalladas, se ha experimentado con su inclusión en forma de narra-

tivas, reales o producidas artificialmente, en las instrucciones dadas a los LLMs. La técnica ha tenido éxito y ha mejorado significativamente el rendimiento de las consultas originales, ya fuera al utilizar un sistema de búsqueda simple como uno de los modelos de re-ranking más potentes encontrados para esta tarea.

En tercer lugar, se ha abordado la predicción de la calidad de consultas en cuanto a la presencia de documentos *harmful* en sus resultados de búsqueda, pero con anterioridad a su ejecución (lo cual tiene importantes aplicaciones, como decidir si es conveniente reformularla). Por primera vez, las técnicas preexistentes para la predicción de la RI de consultas se han evaluado bajo la óptica de la desinformación sanitaria, y nuestros resultados demuestran su eficacia a este respecto. Además, hemos diseñado nuevos métodos que, usando LLMs, mejoran los mencionados predictores tradicionales y logran resultados estadísticamente significativos.

Concluimos que los datos recogidos, sumados a su análisis y a la revisión previa de la literatura relacionada, dan alcance a los objetivos específicos propuestos. Con ello, se cumple también el objetivo principal del proyecto, al aportar tanto un estudio de la aplicabilidad de técnicas preexistentes a la detección y reducción de la desinformación en resultados de búsqueda, como nuevas técnicas que las optimizan y mejoran sustancialmente a través del uso de LLMs.

Por otro lado, hemos abierto diferentes posibilidades para líneas de investigación futuras, entre las que destacamos las siguientes ideas:

- Estudiar las características de las consultas eficaces para el ámbito de la salud y explotarlas para optimizar nuestras técnicas.
- Diseñar estrategias para combinar los resultados de búsqueda de las diferentes variantes de consultas. Por ejemplo, sería interesante probar técnicas de fusión de rankings (*rank fusion*), ya sean supervisadas o no supervisadas.
- Explorar cuál es el número de consultas alternativas que resulta más beneficioso generar para cada consulta, con podría contribuir a mitigar el coste económico del uso de LLMs si se reduce el número de tokens generados.
- Analizar el efecto del tamaño de los rankings de documentos sobre métricas de RI y de desinformación, y estudiar cómo ajustarlo dinámicamente.
- Elaborar un nuevo método de predicción de desinformación que combine técnicas basadas en LLMs con predictores clásicos (por ejemplo, interpolando sus puntuaciones), con la idea de aprovechar sus respectivas fortalezas.
- Examinar el impacto de características de las colecciones de prueba, como la proporción de documentos *harmful*, sobre los métodos probados.

Apéndice A

Manuales Técnicos

En este apéndice se describe la estructura de ficheros del código empleado en el proyecto, así como dónde están disponibles los recursos necesarios para la reproducción de los experimentos.

Estos materiales se incluyen en dos repositorios separados de GitHub. El primero contiene el código y los recursos empleados para llevar a cabo la comparativa de sistemas de búsqueda y la generación de consultas alternativas, y sirve también a modo de repositorio para el artículo corto que se publicó en relación a estos experimentos. El segundo recoge todo lo necesario para los experimentos de predicción de desinformación en consultas. Se ha decidido mantenerlos separados para preservar la integridad del material correspondiente al artículo publicado.

A.1. Repositorio *Generating-Effective-Health-Queries*

En esta sección describimos el repositorio correspondiente a los experimentos de comparación de sistemas de búsqueda y generación de consultas alternativas, disponible a través de Github, en xianacarrera/Generating-Effective-Health-Queries.

La organización de su estructura de ficheros es la siguiente, donde indicamos con puntos suspensivos estructuras de ficheros análogas a las de directorios del mismo nivel ya mostrados:

```
Generating-Effective-Health-Queries/
└── TREC_2020_BEIR/
```

```
├── TREC_2021_BEIR/
├── TREC_2022_BEIR/
├── CLEF/
├── alternative_queries/
│   ├── 2020/
│   │   ├── gpt/
│   │   │   ├── NC1/
│   │   │   ├── RNC1/
│   │   │   ├── orig_C1/
│   │   │   ├── orig_NC1/
│   │   │   ├── orig_RC1/
│   │   │   └── orig_RNC1/
│   │   └── llama/
│   │       └── ...
│   ├── 2021/
│   │   └── ...
│   ├── 2022/
│   │   └── ...
│   └── clef/
│       └── ...
├── generated_narratives/
│   ├── gpt_narratives/
│   ├── llama_narratives/
│   └── preliminary_tests/
└── rankings/
    ├── rankings_2020_gpt/
    ├── rankings_2020_llama/
    ├── rankings_2021_gpt/
    ├── rankings_2021_llama/
    ├── rankings_2022_gpt/
    ├── rankings_2022_llama/
    ├── rankings_clef_gpt/
    └── rankings_clef_llama/
```

Los subdirectorios `TREC_2020_BEIR/`, `TREC_2021_BEIR/`, `TREC_2022_BEIR/` y `CLEF/` contienen los datos originales de las respectivas colecciones de prueba y diferentes ficheros con la misma información convertida a diferentes formatos (por ejemplo, jsonl y tsv, que son los formatos requeridos por la librería BEIR para cargar conjuntos de datos manualmente).

La naturaleza no determinística de los LLMs impide reproducir de forma exacta los experimentos realizados con ellos. Para solventar esta situación y permitir que se comprueben los resultados obtenidos, hemos incluido en el reposi-

torio todas las generaciones obtenidas a través del uso de LLMs: narrativas, en `generated_narratives/` y consultas alternativas, en `alternative_queries/`, un directorio organizado por colección, modelo de LLM y configuración de la instrucción empleada. Las configuraciones aparecen también en numerosos nombres de ficheros y siguen el siguiente formato:

- Se ha incluido un rol solo cuando se indica `R`.
- Se ha incluido una narrativa solo cuando se indica `N`.
- Si no se indica `orig` y aparece `N`, se están usando narrativas sintéticas.
- El tipo de *chain-of-thought* depende del número que aparece a continuación de `C` (0, 1 o 2).

En el directorio `rankings/` incluimos los resultados de búsqueda de BM25 y MiniLM-L-12-v2 para las consultas originales de las colecciones de prueba y para sus alternativas.

Además, en la raíz del directorio se encuentran también los diferentes programas y archivos de configuración necesarios para el proyecto, que son:

- `baseline_bm25.py`: ejecuta BM25 sobre todo el corpus de una de las colecciones de prueba.
- `bm25_corpus_creator.py`: ejecuta BM25 sobre todo un corpus y crea un nuevo directorio con el top 1000 de los documentos recuperados para cada consulta de la colección correspondiente al corpus.
- `sparse.py`: permite ejecutar los modelos dispersos SPARTA y SPLADE sobre el top 1000 de los resultados de BM25.
- `dense.py`: permite ejecutar modelos densos (DPR, ANCE y TAS-B) sobre el top 1000 de los resultados de BM25.
- `reranker.py`: permite ejecutar modelos de re-ranking (ELECTRA-base, MiniLM, TinyBERT y MonoT5) sobre un cierto corte del top 1000 de los resultados de BM25.
- `docT5query_corpus_creator.py`: permite expandir los documentos del top 1000 dado por BM25 con DocT5query.
- `beir_helper.py`: Programa con funciones auxiliares de lectura y escritura de ficheros y limpieza del HTML de documentos. Es utilizado por `baseline_bm25.py`, `sparse.py`, `dense.py`, `reranker.py` y por `docT5query_corpus_creator.py`.

- `llm_connector.py`: Permite realizar generaciones de narrativas y consultas con GPT-4 o LLaMA3.
- `compatibility.py`: Calcula la *compatibility* de un ranking obtenido experimentalmente con respecto a un ranking de resultados ideales. En particular, con este programa se puede calcular las medidas de *compatibility helpful* y de *compatibility harmful*.
- `config.ini.sample`: Ejemplo de configuración de las variables de entorno que determinan la ejecución de los programas.

Asimismo, en la raíz del proyecto se incluye también una licencia (`LICENSE`), una descripción completa del repositorio (`README.md`), un archivo con el texto completo de las instrucciones usadas para los LLMs (`prompts.txt`) y un fichero con los requerimientos de librerías que hacen falta para ejecutar el código (`environment.yml`).

A.2. Repositorio *Query-Misinformation-Prediction*

Esta sección describe los contenidos del repositorio con el código y archivos dedicados al estudio de la predicción de desinformación en consultas. Está disponible a través de Github, en xianacarrera/Query-Misinformation-Prediction.

La estructura del repositorio es la siguiente:

```
Query-Misinformation-Prediction/
├── CLEF_v2/
├── compatibility_results/
├── confidences_query_quality_classifier/
├── controversy_results/
├── harmful_at_k/
├── ndcg_harmful_only_results/
├── topics/
├── traditional_predictors_results/
├── TREC_2020_BEIR/
├── TREC_2021_BEIR/
└── TREC_2022_BEIR/
```

Como en el caso del repositorio *Generating-Effective-Health-Queries*, el directorio `CLEF_v2` contiene los archivos correspondientes a la colección de prueba

CLEF IR 2016, pero esta vez filtrando las consultas y *qrels* para mantener solo aquellas correspondientes a usuarios/as no especializados y empleando la asignación de la ecuación (3.2) para calcular el orden de prioridad en base a la puntuación de fiabilidad (en lugar de la ecuación (3.1), que fue la empleada en *Generating-Effective-Health-Queries*). Se incluyen también los datos de las colecciones de TREC HM en sus respectivos directorios, pero, por comodidad, las consultas se encuentran también recogidas bajo `topics/`.

Las predicciones de GPT-4 para cada consulta de cada colección se incluyen en `controversy_results/`, dado que, por la naturaleza no determinística de los LLMs, sus generaciones no podrían repetirse de forma exacta. Además, por completitud incluimos también las puntuaciones dadas por predictores tradicionales y por el clasificador Query-Quality-Classifier en los directorios `confidences_query_quality_classifier/` y `traditional_predictors_results/`, respectivamente, si bien hacemos notar que estos resultados sí son deterministicos y pueden ser reproducidos.

Para facilitar el cálculo de las métricas de correlación, se incluyen también las puntuaciones de *compatibility* obtenidas para cada consulta de cada colección con BM25 y MiniLM-L-12-v2 en el directorio `compatibility_results/`. Adicionalmente, se proporcionan los resultados del conteo de documentos *harmful* y del cálculo de NDCG@K con ellos para diferentes umbrales, medidas que habían sido planteadas para la evaluación, pero que finalmente se descartaron por ser poco informativas. Estas están en `harmful_at_k/` y `ndcg_harmful_only_results/`, respectivamente.

Por otro lado, en la raíz se incluyen los siguientes archivos de código:

- `qpp_metrics.py`: Programa con nuestra implementación de técnicas de QPP tradicionales (avg IDF, max IDF, avg SCQ, max SCQ, avg ICTF, SCS).
- `query_quality_classifier.py`: Programa que ejecuta Query-Quality- Classifier a través de Hugging Face y calcula una predicción de desinformación en base a la confianza que este reporta.
- `chatgpt.py`: Una versión actualizada del programa `llm_connector.py` de *Generating-Effective-Health-Queries* que añade la funcionalidad necesaria para obtener puntuaciones de controversia de consultas.

En la raíz hay también una licencia (`LICENSE`), una descripción del repositorio (`README.md`), un fichero con el texto completo de las instrucciones de controversia usadas (`controversy_prompts.txt`), un ejemplo de archivo de configuración (`config.ini.sample`, que solo necesita una clave de la API de GPT-4), los requerimientos de librerías necesarias (en `environment_qmp.yml` y `environment_`

`qppmetrics.yml`) y libretas de Jupyter de evaluación y análisis de los resultados (`pre_qpp_analysis_comp.ipynb`, `controversy_analysis_factors.ipynb` y `classifier_analysis.ipynb`).

Apéndice B

Manuales de Usuario

En este apéndice se incluye la información necesaria para ejecutar el código e instalar las dependencias necesarias para ello.

B.1. Configuración del Entorno de Ejecución

Para facilitar la gestión de los diversos entornos virtuales que se necesitarán, hemos optado por utilizar el gestor de paquetes Anaconda. Aunque este es un paso opcional y existen otras alternativas para la gestión de paquetes, recomendamos su uso por simplificar notablemente este tipo de tareas.

Indicamos a continuación los pasos a seguir para instalar Miniconda, una alternativa ligera de Anaconda, en un equipo de arquitectura de 64 bits con sistema operativo Linux:

1. Abrir una terminal en la carpeta donde se desea instalar el programa.
2. Descargar el script de instalación:

```
$> wget https://repo.anaconda.com/miniconda/
Miniconda3-latest-Linux-x86_64.sh
```

3. Ejecutar el script de instalación a través de bash.

```
$> bash Miniconda3-latest-Linux-x86_64.sh
```

4. Cerrar y volver a abrir la terminal para que los cambios hagan efecto.

5. Comprobar que, en el *prompt* de la terminal, se indica ahora "(base)" antes del nombre del/de la usuario/a. Este es el nombre del entorno predeterminado de Anaconda.

6. Comprobar que la instalación ha tenido exito probando el comando `conda list`, que muestra una lista de los paquetes instalados en el entorno actual.

```
$> conda list
```

Para ejecutar el código relativo a los experimentos de comparación de sistemas de búsqueda y generación de consultas alternativas, que está disponible en el repositorio de GitHub xianacarrera/Generating-Effective-Health-Queries, facilitamos la creación de un entorno adecuado a través de un archivo `environment.yml` con los requerimientos de librerías. Para crear dicho entorno, se pueden seguir los siguientes pasos:

1. Clonar el repositorio:

```
$> git clone https://github.com/xianacarrera/
    Generating-Effective-Health-Queries
```

2. Crear un nuevo entorno virtual a partir de `environment.yml`, cuyo nombre por defecto será `healquery`:

```
$> conda env create -f environment.yml
```

3. Activar el entorno:

```
$> conda activate healquery
```

A través de `healquery` se pueden ejecutar los programas de *Generating-Effective-Health-Queries*. Destacamos que usamos una versión modificada manualmente de la clase `GenericDataLoader` del archivo `data_loader.py` de la librería BEIR, que nos permite cargar colecciones de datos sin pasar todo su corpus a través de un archivo jsonl (que no es viable para nuestro proyecto por el gran tamaño de los corpus empleados). Damos a continuación el código a sustituir:

```
class GenericDataLoader:

    def __init__(self, data_folder: str = None, prefix: str = None,
    corpus_file: str = "corpus.jsonl", query_file: str = "queries.
    jsonl", qrels_folder: str = "qrels", qrels_file: str = "",
    use_corpus: bool = True):
        self.corpus = {}
        self.queries = {}
        self.qrels = {}
        self.use_corpus = use_corpus

        if prefix:
            query_file = prefix + "-" + query_file
            qrels_folder = prefix + "-" + qrels_folder

        if self.use_corpus:
            self.corpus_file = os.path.join(data_folder, corpus_file)
            if data_folder else corpus_file
        self.query_file = os.path.join(data_folder, query_file) if
        data_folder else query_file
        self.qrels_folder = os.path.join(data_folder, qrels_folder)
        if data_folder else None
        self.qrels_file = qrels_file

    @staticmethod
    def check(fIn: str, ext: str):
        if not os.path.exists(fIn):
            raise ValueError("File {} not present! Please provide
            accurate file.".format(fIn))

        if not fIn.endswith(ext):
            raise ValueError("File {} must be present with extension
            {}".format(fIn, ext))

    def load_custom(self) -> Tuple[Dict[str, Dict
    [str, str]], Dict[str, str], Dict[str, Dict[str, int]]]:

        if self.use_corpus:
            self.check(fIn=self.corpus_file, ext="jsonl")
        self.check(fIn=self.query_file, ext="jsonl")
        self.check(fIn=self.qrels_file, ext="tsv")
```

```
        if self.use_corpus and not len(self.corpus):
            logger.info("Loading Corpus...")
            self._load_corpus()
            logger.info("Loaded %d Documents.", len(self.corpus))
            logger.info("Doc Example: %s", list(self.corpus.values
            ())[0])

        if not len(self.queries):
            logger.info("Loading Queries...")
            self._load_queries()

        if os.path.exists(self.qrels_file):
            self._load_qrels()
            self.queries = {qid: self.queries[qid] for qid in self.
            qrels}
            logger.info("Loaded %d Queries.", len(self.queries))
            logger.info("Query Example: %s", list(self.queries.values
            ())
            [0])

        return self.corpus, self.queries, self.qrels

    def load(self, split="test") -> Tuple[Dict[str, Dict[str, str]],
    Dict[str, str], Dict[str, Dict[str, int]]]:

        self.qrels_file = os.path.join(self.qrels_folder, split + ".
        tsv")
        self.check(fIn=self.corpus_file, ext="jsonl")
        self.check(fIn=self.query_file, ext="jsonl")
        self.check(fIn=self.qrels_file, ext="tsv")

        if not len(self.corpus):
            logger.info("Loading Corpus...")
            self._load_corpus()
            logger.info("Loaded %d %s Documents.", len(self.corpus),
            split.upper())
            logger.info("Doc Example: %s", list(self.corpus.values
            ())[0])

        if not len(self.queries):
            logger.info("Loading Queries...")
            self._load_queries()
```

```
        if os.path.exists(self.qrels_file):
            self._load_qrels()
            self.queries = {qid: self.queries[qid] for qid in self
            .qrels}
            logger.info("Loaded %d %s Queries.", len(self.queries),
            split.upper())
            logger.info("Query Example: %s", list(self.queries.values
            ())[0])

        return self.corpus, self.queries, self.qrels

    def load_corpus(self) -> Dict[str, Dict[str, str]]:

        self.check(fIn=self.corpus_file, ext="jsonl")

        if not len(self.corpus):
            logger.info("Loading Corpus...")
            self._load_corpus()
            logger.info("Loaded %d Documents.", len(self.corpus))
            logger.info("Doc Example: %s", list(self.corpus.values())
            [0])

        return self.corpus

    def _load_corpus(self):

        num_lines = sum(1 for i in open(self.corpus_file, 'rb'))
        with open(self.corpus_file, encoding='utf8') as fIn:
            for line in tqdm(fIn, total=num_lines):
                line = json.loads(line)
                self.corpus[line.get("_id")] = {
                    "text": line.get("text"),
                    "title": line.get("title"),
                }

    def _load_queries(self):

        with open(self.query_file, encoding='utf8') as fIn:
            for line in fIn:
                line = json.loads(line)
                self.queries[line.get("_id")] = line.get("text")

    def _load_qrels(self):
```

```
reader = csv.reader(open(self.qrels_file, encoding="utf-8"),
                    delimiter="\t", quoting=csv.QUOTE_MINIMAL)
next(reader)

for id, row in enumerate(reader):
    query_id, corpus_id, score = row[0], row[1], int(row[2])

    if query_id not in self.qrels:
        self.qrels[query_id] = {corpus_id: score}
    else:
        self.qrels[query_id][corpus_id] = score
```

Con respecto al código de la predicción de desinformación en consultas, disponible en xianacarrera/Query-Misinformation-Prediction, se necesita la creación de dos entornos, para poder mantener versiones de Pyserini y Transfomers correspondientes a versiones de Python diferentes. La elección de una versión de Pyserini relativamente antigua se debe a problemas de compatibilidad entre alternativas más actualizadas con librerías que se usaban habitualmente para programas relacionados, como BEIR, en el contexto del clúster `ctcomp3` del CiTIUS.

Para crear estos entornos, se pueden seguir las siguientes instrucciones:

1. Clonar el repositorio:

```
$> git clone https://github.com/xianacarrera/
    Query-Misinformation-Prediction
```

2. Crear los dos nuevos entornos a patir de los archivos `environment_qmp.yml` y `environment_qppmetrics.yml`, cuyos nombres predeterminados son `qmp` y `qppmetrics`, respectivamente:

```
$> conda env create -f environment_qmp.yml
$> conda env create -f environment_qppmetrics.yml
```

3. Activar uno u otro entorno.

```
$> conda activate qmp
$> conda activate qppmetrics
```

El programa *qpp_metrics.py*, que emplea Pyserini, requiere el uso de `qppmetrics`. El resto de programas se apoyan en las librerías de `qmp`.

B.2. Ejecución de los programas

A continuación, explicamos qué aspectos deben tenerse en cuenta para ejecutar los programas de cada repositorio.

B.2.1. De *Generating-Effective-Health-Queries*

Para ejecutar los modelos de búsqueda de BEIR a través de los programas `sparse.py`, `dense.py`, `reranker.py` y `docT5query_corpus_creator.py`, se debe haber ejecutado antes `bm25_corpus_creator.py`, que crea un top 1000 de resultados de BM25 a partir de todo el corpus. Cada corpus debe haber sido indexado previamente con Pyserini, con una instrucción análoga a la siguiente:

```
$> python -m pyserini.index \
        --input integrations/resources/sample_collection_jsonl \
        --collection JsonCollection \
        --generator DefaultLuceneDocumentGenerator \
        --index indexes/sample_collection_jsonl \
        --threads 1 \
        --storePositions --storeDocvectors --storeRaw
```

Cada uno de estos programas genera un ranking de resultados y escribe en un fichero las métricas de RI registradas a través de BEIR.

El programa `llm_connector.py` se puede ejecutar directamente y presenta un menú con las diferentes funcionalidades de generación de consultas y narrativas disponibles al/a la usuario/a.

Las opciones de estos programas (entre otros, la clave de API de GPT-4; las dirección de los directorios donde están los índices de Lucene, las consultas, los qrels y los tops 1000 de BM25; la dirección de los archivos donde se escribirá la salida; y los modelos a emplear) se deben configurar a través del fichero `config.ini.sample`, que incluye una descripción del significado y formato a seguir por cada uno de los parámetros.

B.2.2. De *Query-Misinformation-Prediction*

El programa `qpp_metrics.py`, que calcula puntuaciones de predictores de QPP clásicos, ha de ser ejecutado sobre los índices de Lucene de los corpus, que tienen que ser creados con la versión de Pyserini del entorno de Anaconda `qppmetrics`. Requiere dos argumentos: el nombre de la colección a emplear y la métrica a calcular. En el propio programa se incluyen *mappings* de ejemplo con correspondencias entre nombres de corpus y las direcciones de sus consultas e índices, que han de ser modificados con los valores apropiados.

El programa `chapgpt.py`, que permite generar puntuaciones de controversia de consultas, sigue la misma estructura que `llm_connector.py` y también presenta un menú para la elección de la funcionalidad.

Por último, `query_quality_classifier.py` permite ejecutar el modelo Query-Quality-Classifier de Hugging Face. No requiere consideraciones particulares.

Apéndice C

Información Complementaria

El presente apéndice reúne datos y recursos que amplían el contenido del trabajo y que, por motivos de espacio y/o de la fluidez y continuidad de lectura, fueron excluidos del cuerpo principal del trabajo. En concreto, incluimos aquí el texto completo de las instrucciones empleadas para la generación de narrativas con LLMs, la especificación de modelos e hiperparámetros de los sistemas de búsqueda empleados a lo largo del trabajo y los resultados del experimento de generación de consultas alternativas empleando LLaMA3 en lugar de GPT-4.

C.1. Instrucciones para la Generación de Narrativas Sintéticas

Como se explicó en la Subsección 3.4.1, se probaron diferentes estilos de instrucciones al generar narrativas sintéticas con LLMs para consultas. Con la misma notación para estas variantes que la empleada en dicho apartado, y denotando por [consulta] los fragmentos a sustituir por una consulta de entrada, las instrucciones completas son:

(a) *Given the query* [consulta]*, write a narrative detailing the information need and describing the characteristics of helpful and harmful documents. Write one paragraph and do not repeat the query in your answer.*

(b) *Given the query* [consulta]*, write a narrative detailing the information need and describing the characteristics of helpful and harmful documents using the standard TREC format for narratives. Write one paragraph and do not repeat the query in your answer.*

(c) *Given the query* [consulta], *write a narrative that describes its information need in more detail. Use the standard TREC format for narratives, with the following style guidelines:*

- *The narrative should have the following structure:*
  - *Begin by describing a specific health claim or rumor, such as health remedies or conspiracy theories.*
  - *Then, provide context, such as sources of misinformation or typical misconceptions.*
  - *Finally, outline specific criteria for helpful documents (those providing truthful and safe instructions) and harmful documents (those that mislead or fail to clarify risks).*
- *The voice should be:*
  - *Objective and neutral, delivering information without bias or emotional language. Focus on a clear presentation of facts.*
  - *Authoritative and factual, providing scientifically grounded statements, particularly when clarifying health misinformation.*
  - *Instructional, offering guidance on what readers should consider reliable information versus misleading information.*
- *The tone should be:*
  - *Informative and cautious, in a way that prevents misinformation by carefully explaining what constitutes helpful versus harmful information.*
  - *Calm and reassuring, addressing potentially anxiety-indu- cing topics in a composed manner to reduce panic or confusion.*
  - *Clear-cut and decisive, distinguishing between helpful and harmful documents in a straightforward, definitive way to reduce ambiguity.*
- *Use a language style that is:*
  - *Plain and accessible, with simple language that makes the content understandable to a wide audience.*
  - *Concise and direct. Each narrative should avoid unnecessary detail, focusing on the essentials of what is helpful or harmful.*
  - *Predictable. It should follow a consistent pattern that helps readers quickly differentiate between reliable and unreliable information.*

Write a complete narrative for the query in a single paragraph. Do not include any other information and do not repeat the query in your answer.

C.2. Características de los Sistemas de Búsqueda Empleados

Reportamos en el Cuadro C.1 las versiones e hiperparámetros fijados para cada uno de los sistemas de búsqueda ejecutados a través de la librería BEIR. Esta información es relevante, sobre todo, para la Sección 4.2.

Cuadro C.1: Información de las versiones e hiperparámetros de los sistemas de búsqueda comparados. Para docT5query también se probó a generar 5 consultas por pasaje, pero al no apreciar diferencias en los resultados, en este trabajo reportamos solo `ques_per_passage` = 3.

Modelo	`batch_size`	`score_function`	Otros parámetros	Implementación
SPARTA	128	cosine	—	https://huggingface.co/BeIR/sparta-msmarco-distilbert-base-v1
SPLADE	128	dot	—	https://huggingface.co/naver/splade_v2_max
docT5query	80	—	`ques_per_passage`= 3, `use_fast` = False, `max_length`= 64, `top_p`= 0.95, `top_k`= 10	https://huggingface.co/castorini/doc2query-t5-base-msmarco
DPR	128	dot	`sep` = " [SEP] "	query: https://huggingface.co/sentence-transformers/facebook-dpr-question_encoder-multiset-base, context: https://huggingface.co/sentence-transformers/facebook-dpr-ctx_encoder-multiset-base
ANCE	128	dot	`corpus_chunk_size`= 50000	https://huggingface.co/sentence-transformers/msmarco-roberta-base-ance-firstp
TAS-B	256	cosine	`corpus_chunk_size` = 512 ∗ 9999	https://huggingface.co/sentence-transformers/msmarco-distilbert-base-tas-b
ELECTRA-base	128	—	—	https://huggingface.co/cross-encoder/ms-marco-electra-base
MiniLM-L-4-v2, MiniLM-L-6-v2, MiniLM-L-12-v2	128	—	—	MiniLM-L-4-v2: https://huggingface.co/cross-encoder/ms-marco-MiniLM-L4-v2, MiniLM-L-6-v2: https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2, MiniLM-L-12-v2: https://huggingface.co/cross-encoder/ms-marco-MiniLM-L12-v2
TinyBERT-L-2-v2, TinyBERT-L-4, TinyBERT-L-6	128	—	—	TinyBERT-L-2-v2: https://huggingface.co/cross-encoder/ms-marco-TinyBERT-L2-v2, TinyBERT-L-4: https://huggingface.co/cross-encoder/ms-marco-TinyBERT-L4, TinyBERT-L-6: https://huggingface.co/cross-encoder/ms-marco-TinyBERT-L6
MonoT5 (base, large & base-med)	128	—	`token_false`="_false", `token_true`="_true"	base: https://huggingface.co/castorini/monot5-base-msmarco, large: https://huggingface.co/castorini/monot5-large-msmarco, base-med: https://huggingface.co/castorini/monot5-base-med-msmarco

C.3. Resultados de la Generación de Consultas Alternativas con LLaMA3

En el Cuadro C.2 incluimos los resultados cuantitativos obtenidos en las pruebas de generación de consultas alternativas con LLaMA3. Es especialmente interesante comparar estos con los del Cuadro 4.3 de la Sección 4.3, en el que se pueden observar tendencias muy similares, aunque con un rendimiento ligeramen-

Cuadro C.2: *Compatibility helpful*, *harmful* y *helpful-harmful* para cada configuración posible de *prompting* en el proceso de generación de consultas (parámetros R, N y C y uso de narrativa real o sintética), empleando LLaMA3. Para cada bloque, conjunto de datos y métrica, se destaca el mejor resultado en negrita. Las puntuaciones de *compatibility helpful* y *harmful* se marcan con * cuando la mejora sobre la referencia es estadísticamente significativa, empleando el test de Wilcoxon unilateral con p-valor $< .05$.

Modelo	Consultas			TREC HM 2020			TREC HM 2021		
	R	**N**	**C**	Help	Harm	Help - Harm	Help	Harm	Help - Harm
BM25	originales			**.214**	.047	**.167**	**.129**	.145	-.016
sin narrativa	0	0	1	.157	.043	.114	.062	.072*	-.010
	1	0	1	.149	.038	.121	.061	.067*	-.006
narrativa real	0	1	1	.178	.034	.144	.083	.062*	**.021**
	1	1	1	.206	.045	.161	.074	.061*	.013
narrativa sintética	0	1	1	.139	**.031**	.108	.060	.060*	.000
	1	1	1	.160	.038	.122	.047	**.053***	-.006
MiniLM-12	originales			.226	**.078**	.148	.132	.136	-.004
sin narrativa	0	0	1	.269*	.097	.172	.136	.126	**.010**
	1	0	1	.276*	.091	.185	.131	**.125**	.006
narrativa real	0	1	1	**.306***	.082	**.224**	.135	.139	-.004
	1	1	1	.304*	.093	.210	**.138**	.136	.002
narrativa sintética	0	1	1	.279*	.083	.196	.137	.134	.003
	1	1	1	.272*	.093	.179	.133	.133	.000

Model	Consultas			TREC HM 2022			CLEF_v1		
	R	**N**	**C**	Help	Harm	Help - Harm	Help	Harm	Help - Harm
BM25	originales			**.173**	.144	**.029**	**.101**	.272	-.172
sin narrativa	0	0	1	.062	.050*	.012	.070	.122*	-.052
	1	0	1	.064	.061*	.003	.067	.128*	-.061
narrativa real	0	1	1	.049	.052*	-.003	——	——	——
	1	1	1	.059	.053*	.006	——	——	——
narrativa sintética	0	1	1	.046	**.047***	-.001	.071	**.106***	**-.035**
	1	1	1	.050	.048*	.002	.068	.113*	-.045
MiniLM-12	originales			.179	.131	.048	.095	.211	-.116
sin narrativa	0	0	1	.182	.133	.049	.096	**.185***	-.089
	1	0	1	.181	.134	.047	.095	.189*	-.094
narrativa real	0	1	1	**.186**	.128	**.058**	——	——	——
	1	1	1	.184	.130	.054	——	——	——
narrativa sintética	0	1	1	.176	**.123**	.053	.097*	.186*	-.089
	1	1	1	.180	.124	.056	**.098***	.185*	**-.087**

te superior. Hacemos notar que cierta parte del aumento en *compatibility helpful* y *compatibility harmful* para las consultas de GPT-4 puede atribuirse a un mejor rendimiento de estas en la RI, pero que el hecho de que la diferencia entre ambas métricas, *compatibility helpful-harmful*, sea también superior para GPT-4 indica que su generación de consultas también es más eficaz a la hora de reducir la desinformación en los resultados de búsqueda.

Bibliografía

[1] G. Eysenbach, "Infodemiology: the epidemiology of (mis)information," *The American Journal of Medicine*, vol. 113, no. 9, pp. 763–765, 2002.

[2] Reuters Institute, "Digital news report, 2024." https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2024, 2024. Consultado el 26 de febrero de 2025.

[3] F. A. Pogacar, A. Ghenai, M. D. Smucker, and C. L. Clarke, "The positive and negative influence of search results on people's decisions about the efficacy of medical treatments," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, (New York, NY, USA), pp. 209–216, Association for Computing Machinery, 2017.

[4] N. Vigdor, "Man fatally poisons himself while self-medicating for coronavirus, doctor says," *The New York Times*, 2020. https://www.nytimes.com/2020/03/24/us/chloroquine-poisoning-coronavirus.html. Consultado el 8 de mayo de 2025.

[5] M. S. Islam, T. Sarkar, S. H. Khan, A.-H. M. Kamal, S. M. M. Hasan, A. Kabir, D. Yeasmin, M. A. Islam, K. I. A. Chowdhury, K. S. Anwar, A. A. Chughtai, and H. Seale, "Covid-19–related infodemic and its impact on public health: A global social media analysis," *The American Journal of Tropical Medicine and Hygiene*, vol. 103, no. 4, pp. 1621–1629, 2020.

[6] World Health Organization, *Managing Epidemics: Key Facts about Major Deadly Diseases*. Luxembourg: World Health Organization, 2018.

[7] S. Fox, "Health topics: 80 % of internet users look for health information online." *Pew Internet & American Life Project*, 2011.

[8] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI*, 2019.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Association for Computational Linguistics, 2019.

[11] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "Pre-trained language models for text generation: A survey," *ACM Comput. Surv.*, vol. 56, no. 9, 2024.

[12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

[13] K. M. Griffiths, T. T. Tang, D. Hawking, and H. Christensen, "Automated assessment of the quality of depression websites," *J Med Internet Res*, vol. 7, no. 5, p. e59, 2005.

[14] Y. Shen, L. Heacock, J. Elias, K. Hentel, B. Reig, G. Shih, and L. Moy, "ChatGPT and other large language models are double-edged swords," *Radiology*, vol. 307, no. 2, 2023.

[15] K. A. Hambarde and H. Proença, "Information retrieval: Recent advances and beyond," *IEEE Access*, vol. 11, pp. 76581–76604, 2023.

[16] B. Croft, D. Metzler, and T. Strohman, *Search Engines: Information Retrieval in Practice*. USA: Addison-Wesley Publishing Company, 1st ed., 2009.

[17] A. Yates, R. Nogueira, and J. Lin, "Pretrained transformers for text ranking: BERT and beyond," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorials* (G. Kondrak, K. Bontcheva, and D. Gillick, eds.), pp. 1–4, Association for Computational Linguistics, 2021.

[18] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[19] F. Du, J. Zhang, J. Hu, and R. Fei, "Discriminative multi-modal deep generative models," *Knowledge-Based Systems*, vol. 173, pp. 74–82, 2019.

[20] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[21] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, *et al.*, "Okapi at TREC-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[22] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis.," *Journal of the American Society for Information Science 41*, pp. 391–407, 1990.

[23] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," in *Advances in Neural Information Processing Systems* (T. Dietterich, S. Becker, and Z. Ghahramani, eds.), vol. 14, MIT Press, 2001.

[24] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, (New York, NY, USA), pp. 89–96, Association for Computing Machinery, 2005.

[25] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, (New York, NY, USA), pp. 55–64, Association for Computing Machinery, 2016.

[26] B. Mitra, F. Diaz, and N. Craswell, "Learning to match using local and distributed representations of text for web search," in *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, (Republic and Canton of Geneva, CHE), pp. 1291–1299, International World Wide Web Conferences Steering Committee, 2017.

[27] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant, "Towards effective and efficient sparse neural information retrieval," *ACM Trans. Inf. Syst.*, vol. 42, no. 5, 2024.

[28] Z. Dai and J. Callan, "Context-aware sentence/passage term importance estimation for first stage retrieval," *arXiv preprint arXiv:1910.10687*, 2019.

[29] J. Lin and X. Ma, "A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques," *arXiv preprint arXiv:2106.14807*, 2021.

[30] R. Nogueira, W. Yang, J. Lin, and K. Cho, "Document expansion by query prediction," *arXiv preprint arXiv:1904.08375*, 2019.

[31] R. Nogueira and J. Lin, "From doc2query to docTTTTTquery," 2019.

[32] H. Zamani, M. Dehghani, W. B. Croft, E. G. Learned-Miller, and J. Kamps, "From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing," *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[33] K.-R. Jang, J. Kang, G. Hong, S.-H. Myaeng, J. Park, T. Yoon, and H. Seo, "Ultra-high dimensional sparse representations with binarization for efficient text retrieval," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, eds.), (Online and Punta Cana, Dominican Republic), pp. 1016–1029, Association for Computational Linguistics, 2021.

[34] I. Yamada, A. Asai, and H. Hajishirzi, "Efficient passage retrieval with hashing for open-domain question answering," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (C. Zong, F. Xia, W. Li, and R. Navigli, eds.), pp. 979–986, Association for Computational Linguistics, 2021.

[35] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[36] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," *arXiv preprint arXiv:2007.00808*, 2020.

[37] O. Khattab and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over bert," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, (New York, NY, USA), pp. 39–48, Association for Computing Machinery, 2020.

[38] F. Scarselli, S. L. Yong, M. Gori, M. Hagenbuchner, A. C. Tsoi, and M. Maggini, "Graph neural networks for ranking web pages," in *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pp. 666–672, 2005.

[39] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, (New York, NY, USA), pp. 793–803, Association for Computing Machinery, 2019.

[40] I. Vulić and M.-F. Moens, "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings," in *Proceedings of the*

38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, (New York, NY, USA), pp. 363–372, Association for Computing Machinery, 2015.

[41] H. Déjean, S. Clinchant, and T. Formal, "A thorough comparison of cross-encoders and LLMs for reranking SPLADE," *arXiv preprint arXiv:2403.10407*, 2024.

[42] L. Azzopardi, "Cognitive biases in search: A review and reflection of cognitive biases in information retrieval," in *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, CHIIR '21, (New York, NY, USA), pp. 27–37, Association for Computing Machinery, 2021.

[43] S. C. Matthews, A. Camacho, P. J. Mills, and J. E. Dimsdale, "The internet for medical information about cancer: Help or hindrance?," *Psychosomatics*, vol. 44, no. 2, pp. 100–103, 2003.

[44] S. L. Price and W. R. Hersh, "Filtering web pages for quality indicators: An empirical approach to finding high quality health information on the world wide web," in *Proceedings of AMIA Symposium*, pp. 911–915, 1999.

[45] P. Sondhi, V. G. V. Vydiswaran, and C. X. Zhai, "Reliability prediction of webpages in the medical domain," in *Proceedings of the 34th European Conference on Advances in Information Retrieval*, ECIR'12, (Berlin, Heidelberg), pp. 219–231, Springer-Verlag, 2012.

[46] M. Fernández-Pichel, D. E. Losada, J. C. Pichel, and D. Elsweiler, "Reliability prediction for health-related content: A replicability study," in *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, (Berlin, Heidelberg), pp. 47–61, Springer-Verlag, 2021.

[47] Y. Zhao, J. Da, and J. Yan, "Detecting health misinformation in online health communities: Incorporating behavioral features into machine learning based approaches," *Information Processing Management*, vol. 58, no. 1, p. 102390, 2021.

[48] R. Pradeep, X. Ma, X. Zhang, H. Cui, R. Xu, R. Nogueira, and J. Lin, "H2oloo at TREC 2020: When all you got is a hammer... deep learning, health misinformation, and precision medicine," in *Proceedings of the 29th Text REtrieval Conference (TREC)*, 2020.

[49] M. Fernández-Pichel, D. E. Losada, and J. C. Pichel, "A multistage retrieval system for health-related misinformation detection," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105211, 2022.

[50] M. Abualsaud, N. Ghelani, H. Zhang, M. D. Smucker, G. V. Cormack, and M. R. Grossman, "A system for efficient high-recall retrieval," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1317–1320, 2018.

[51] M. Abualsaud, K. Ghajar, L. Minh, D. Zhang, I. Chen, M. Smucker, and A. Tahami, "UWaterlooMDS at the TREC 2021 Health Misinformation Track," 2021.

[52] R. Pradeep, X. Ma, R. Nogueira, and J. Lin, "Scientific claim verification with VerT5erini," in *Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis*, (online), pp. 94–103, 2021.

[53] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," 2023.

[54] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, (Red Hook, NY, USA), Curran Associates Inc., 2022.

[55] A. Bacciu, E. Palumbo, A. Damianou, N. Tonellotto, and F. Silvestri, "Generating query recommendations via LLMs," *arXiv preprint arXiv:2405.19749*, 2024.

[56] K. D. Dhole and E. Agichtein, "GenQREnsemble: Zero-shot LLM ensemble prompting fornbsp;generative query reformulation," in *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part III*, (Berlin, Heidelberg), pp. 326–335, Springer-Verlag, 2024.

[57] W. Yao, Y. Wang, Z. Yu, R. Xie, S. Zhang, and W. Ye, "PURE: Aligning LLM via pluggable query reformulation for enhanced helpfulness," in *Findings of the Association for Computational Linguistics: EMNLP 2024* (Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, eds.), (Miami, Florida, USA), pp. 8721–8744, Association for Computational Linguistics, 2024.

[58] L. Wang, N. Yang, and F. Wei, "Query2doc: Query expansion with large language models," in *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[59] I. Mackie, S. Chatterjee, and J. Dalton, "Generative relevance feedback with large language models," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR

'23, (New York, NY, USA), pp. 2026–2031, Association for Computing Machinery, 2023.

[60] P. Thomas, S. Spielman, N. Craswell, and B. Mitra, "Large language models can accurately predict searcher preferences," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, (New York, NY, USA), pp. 1930–1940, Association for Computing Machinery, 2024.

[61] C. Meng, N. Arabzadeh, M. Aliannejadi, and M. de Rijke, "Query performance prediction: From ad-hoc to conversational search," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, (New York, NY, USA), pp. 2583–2593, Association for Computing Machinery, 2023.

[62] M. Li, H. Zhuang, K. Hui, Z. Qin, J. Lin, R. Jagerman, X. Wang, and M. Bendersky, "Can query expansion improve generalization of strong cross-encoder rankers?," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, (New York, NY, USA), pp. 2321–2326, Association for Computing Machinery, 2024.

[63] M. Khodabakhsh, F. Zarrinkalam, and N. Arabzadeh, "BertPE: A BERT-based pre-retrieval estimator for query performance prediction," in *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part III*, (Berlin, Heidelberg), p. 354–363, Springer-Verlag, 2024.

[64] D. Carmel and E. Yom-Tov, "Estimating the query difficulty for information retrieval," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 2, p. 911, 2010.

[65] Y. Zhao, F. Scholer, and Y. Tsegay, "Effective pre-retrieval query performance prediction using similarity and variability evidence," in *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR'08, (Berlin, Heidelberg), pp. 52–64, Springer-Verlag, 2008.

[66] B. He and I. Ounis, "Inferring query performance using pre-retrieval predictors," *Proceedings of SPIRE*, vol. 3246, pp. 43–54, 2004.

[67] G. Faggioli, T. Formal, S. Marchesin, S. Clinchant, N. Ferro, and B. Piwowarski, "Query performance prediction for neural IR: Are we there yet?," in *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part I*, (Berlin, Heidelberg), pp. 232–248, Springer-Verlag, 2023.

[68] N. Arabzadeh, M. Khodabakhsh, and E. Bagheri, "BERT-QPP: Contextualized pre-trained transformers for query performance prediction," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, (New York, NY, USA), pp. 2857–2861, Association for Computing Machinery, 2021.

[69] C. L. A. Clarke, M. Maistro, S. Rizvi, M. D. Smucker, and G. Zuccon, "Overview of the TREC 2020 Health Misinformation Track," 2020.

[70] C. L. A. Clarke, M. Maistro, and M. D. Smucker, "Overview of the TREC 2021 Health Misinformation Track," 2021.

[71] C. L. A. Clarke, M. Maistro, M. Seifikar, and M. D. Smucker, "Overview of the TREC 2022 Health Misinformation Track (notebook)," 2022.

[72] L. Kelly, L. Goeuriot, H. Suominen, A. Névéol, J. Palotti, and G. Zuccon, "Overview of the CLEF eHealth Evaluation Lab 2016," 2016.

[73] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2023.

[74] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[75] Intel, "Procesador Intel Core i7-9700k." https://www.intel.la/content/www/xl/es/products/sku/186604/intel-core-i79700k-processor-12m-cache-up-to-4-90-ghz/specifications.html?wapkw=intel%20core%20i7-9700k%203%2060. Consultado por última vez el 20 de junio de 2025.

[76] NVIDIA, "NVIDIA GeForce GTX 1050." https://www.nvidia.com/es-la/geforce/products/10series/geforce-gtx-1050/. Consultado por última vez el 20 de junio de 2025.

[77] D. Technologies, "Dell EMC PowerEdge R740 - Hoja de especificaciones." https://i.dell.com/sites/csdocuments/Product_Docs/es/PowerEdge-R7525-Spec-Sheet.pdf. Consultado por última vez el 20 de junio de 2025.

[78] Intel, "Procesador Intel Xeon Gold 5220." https://www.intel.la/content/www/xl/es/products/sku/193388/intel-xeon-gold-5220-processor-24-75m-cache-2-20-ghz/specifications.html?wapkw=intel%20xeon%20gold%205220. Consultado por última vez el 20 de junio de 2025.

[79] Intel, "Procesador Intel Xeon Gold 5220r." https://www.intel.la/content/www/xl/es/products/sku/199354/intel-xeon-gold-5220r-processor-35-75m-cache-2-20-ghz/specifications.html?wapkw=intel%20xeon%20gold%205220. Consultado por última vez el 20 de junio de 2025.

[80] D. Technologies, "Dell EMC PowerEdge R840 - Hoja de especificaciones de R840." https://i.dell.com/sites/csdocuments/shared-content_data-sheets_documents/es/la/poweredg-r840-spec-sheet-la.pdf. Consultado por última vez el 20 de junio de 2025.

[81] Intel, "Procesador Intel Xeon Gold 6248." https://www.intel.la/content/www/xl/es/products/sku/192446/intel-xeon-gold-6248-processor-27-5m-cache-2-50-ghz/specifications.html?wapkw=intel%20xeon%20gold%206248. Consultado por última vez el 20 de junio de 2025.

[82] NVIDIA, "NVIDIA V100 Tensor Core GPU." https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf. Consultado por última vez el 20 de junio de 2025.

[83] D. Technologies, "Dell EMC PowerEdge R7525 - Guía técnica." https://i.dell.com/sites/csdocuments/Product_Docs/es/PowerEdge-R7525-Spec-Sheet.pdf. Consultado por última vez el 20 de junio de 2025.

[84] AMD, "Procesador AMD EPYC 7543." https://www.amd.com/es/products/processors/server/epyc/7003-series/amd-epyc-7543.html. Consultado por última vez el 20 de junio de 2025.

[85] NVIDIA, "NVIDIA A100 Tensor Core GPU." https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf. Consultado por última vez el 20 de junio de 2025.

[86] P. Yang, H. Fang, and J. Lin, "Anserini: Enabling the use of Lucene for information retrieval research," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, (New York, NY, USA), pp. 1253–1256, Association for Computing Machinery, 2017.

[87] E. Hatcher and O. Gospodnetic, *Lucene in Action (In Action series)*. USA: Manning Publications Co., 2004.

[88] C. L. A. Clarke, A. Vtyurina, and M. D. Smucker, "Assessing top- preferences," *ACM Trans. Inf. Syst.*, vol. 39, no. 3, 2021.

[89] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[90] T. Zhao, X. Lu, and K. Lee, "SPARTA: Efficient open-domain question answering via sparse transformer matching retrieval," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, eds.), pp. 565–575, Association for Computational Linguistics, 2021.

[91] T. Formal, B. Piwowarski, and S. Clinchant, "SPLADE: Sparse lexical and expansion model for first stage ranking," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2288–2292, 2021.

[92] S. Hofstätter, S.-C. Lin, J.-H. Yang, J. Lin, and A. Hanbury, "Efficiently teaching an effective dense retriever with balanced topic aware sampling," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 113–122, 2021.

[93] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[94] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.

[95] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2020.

[96] R. Nogueira, Z. Jiang, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," *arXiv preprint arXiv:2003.06713*, 2020.

[97] M. Faruqui and D. Das, "Identifying well-formed natural language questions," in *Proc. of EMNLP*, 2018.

[98] A. Saleminezhad, N. Arabzadeh, S. Beheshti, and E. Bagheri, "Context-aware query term difficulty estimation for performance prediction," in *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part IV*, (Berlin, Heidelberg), pp. 30–39, Springer-Verlag, 2024.