Escuela Politécnica Nacional

Facultad de Ingeniería de Sistemas

Ingeniería en Computación

Recuperación de la Información

Prof. Iván Carrera

Estudiantes: Roberth Gancino y Danny Iñaguazo

Fecha: 09/12/2025

Informe Técnico

1. Descripción del proyecto

- **Objetivo:** Diseñar e implementar un sistema de recuperación de información que indexe un conjunto de documentos en texto plano y permita ejecutar consultas de texto libre.
- **Resultados Clave:**
  1. **Indexación de Documentos:** Un sistema capaz de indexar un conjunto de documentos en texto plano.
  2. **Modelos de Recuperación Implementados:**
     - El modelo vectorial con vectores binarios.
     - El modelo vectorial con ponderación TF-IDF.
     - El modelo probabilístico BM25.
  3. **Ejecución de Consultas:** Funcionalidad para ejecutar consultas de texto libre.
  4. **Evaluación de Calidad:** La capacidad de evaluar la calidad de los resultados utilizando métricas estándar como precision y recall.

2. Introducción

El acceso eficiente y la gestión de la información se han convertido en pilares fundamentales en la era digital. La necesidad de localizar documentos relevantes dentro de grandes colecciones de texto plano de manera rápida y precisa exige el diseño e implementación de sistemas de recuperación de información (IR) robustos y bien evaluados.

El presente informe técnico documenta el diseño, la implementación y la evaluación de un Sistema de Recuperación de Información destinado a procesar un corpus de documentos en texto plano. El alcance del proyecto se centra en la construcción de un

índice invertido eficiente, la implementación de tres modelos de recuperación clave (Jaccard binario, Coseno con TF-IDF, y BM25), la provisión de una interfaz de línea de comandos (CLI) básica, y la evaluación rigurosa del rendimiento utilizando métricas estándar como Precision y Recall a nivel de consulta, y MAP (Mean Average Precision) a nivel global.

El objetivo principal es comparar la efectividad de los modelos de recuperación implementados y determinar qué enfoque proporciona el *ranking* de resultados más relevante para las consultas de texto libre ejecutadas sobre el corpus seleccionado.

Este documento está estructurado para detallar, en las secciones siguientes, la descripción del corpus utilizado, las decisiones técnicas clave tomadas durante el diseño, la explicación de los modelos de recuperación implementados, los ejemplos de consultas y resultados, y el análisis exhaustivo de las métricas de evaluación obtenidas.

3. Descripción del Corpus y Procesamiento

3.1. Descripción del Corpus Utilizado

El sistema de recuperación se implementó y evaluó sobre un corpus consolidado a partir de cuatro documentos csv del Medical Q&A Dataset (derivado de MedQuAD). El *corpus* final combina información de múltiples dominios médicos, proporcionando un entorno de prueba variado y exigente.

Los archivos de origen utilizados para la consolidación fueron:

- CancerQA.csv
- Genetic_and_Rare_DiseasesQA.csv
- Diabetes_and_Digestive_and_Kidney_DiseasesQA.csv
- SeniorHealthQA.csv

Característica	Detalle
Consolidación	Los archivos fueron unidos horizontalmente, eliminando la columna `Split`.
Tamaño del Corpus	El *corpus* consolidado cuenta con 8,078 filas y 3 columnas.
Documentos del Corpus	La columna `Answer` es la fuente de texto plano que se utilizó para construir el índice. Cada respuesta individual se considera un documento único.

Documentos relevantes	Los pares relevantes (qrels) se definen implícitamente: un documento (`Answer`) es relevante para una consulta (`Question`) si dicha consulta aparece repetida en la columna `Question`, indicando que la respuesta es la *ground truth* para esa pregunta.
Naturaleza del Texto	Texto plano, técnico, informativo, en **inglés**, sobre salud y enfermedades.

3.2. Construcción del Índice

La construcción del índice se basa en un proceso de preprocesamiento riguroso que se aplica de manera uniforme a los documentos del corpus (columna `Answer`). Este proceso se encapsula en la función `preProcesar(texto)` asegurando la coherencia entre el espacio de indexación y el espacio de consulta, dicha función se replica en cada una de las clases para cada modelo de búsqueda.

- **Función de Preprocesamiento Implementada (`preProcesar`):**

El proceso de tokenización y filtrado se define mediante la siguiente lógica:

1. **Normalización (Minúsculas):** El texto se convierte a minúsculas (`textoMin = texto.lower()`). Esto garantiza que la búsqueda sea insensible a las mayúsculas.
2. **Tokenización:** Se utiliza un tokenizador de NLTK para dividir el texto en una lista de *tokens*.
3. **Filtrado de Tokens:** Se aplican dos criterios de filtrado en el mismo paso:
    a. **Remoción de Puntuación/Numéricos:** Solo se retienen *tokens* que son estrictamente alfabéticos (`token.isalpha()`). Esto elimina números, símbolos y la mayoría de los signos de puntuación, manteniendo el enfoque en el contenido semántico.
    b. **Remoción de Stopwords:** Los *tokens* restantes se comparan con una lista de *stopwords* predefinida y se excluyen los términos de alta frecuencia y baja capacidad discriminatoria.

- **Estructura del Índice Invertido:**

El índice almacena, para cada término resultante del proceso `preProcesar`:

- La lista de *posteo* de los documentos (IDs de la columna `Answer`) donde aparece.

- La **frecuencia del término (tf)** dentro de cada documento para su uso en los modelos TF-IDF y BM25.

Esta metodología garantiza que la indexación sea eficiente y que las representaciones vectoriales y probabilísticas se construyan sobre un vocabulario depurado y normalizado.

4. Diseño e Implementación del Modelo de Recuperación

4.1. Decisiones de Diseño

El sistema de recuperación de información fue diseñado y construido bajo una arquitectura modular en el lenguaje Python, aprovechando su robustez y el vasto ecosistema de librerías para el procesamiento de datos y lenguaje natural.

Aspecto de Diseño	Herramientas y Justificación
Lenguaje y Entorno	Python se eligió por su sencillez, sintaxis clara, y excelente rendimiento en tareas de manipulación de datos y prototipado rápido de algoritmos.
Manipulación de Datos	Pandas y NumPy se utilizaron para la carga, consolidación, y manipulación eficiente de los datos del corpus (los 8,078 documentos), así como para operaciones matriciales necesarias en los modelos vectoriales (TF-IDF y Coseno).
Persistencia del Sistema	`Pickle` se empleó para la serialización y des-serialización de objetos clave, como el Índice Invertido y, potencialmente, las matrices TF-IDF, lo que permite cargar rápidamente el sistema sin necesidad de re-indexar el corpus completo en cada ejecución.
Interfaz de Usuario (CLI)	Textual se seleccionó esta librería para implementar la Interfaz de Línea de Comandos (CLI) del sistema. Textual facilita la creación de interfaces más interactivas y visualmente organizadas dentro de la terminal, superando las limitaciones de una CLI tradicional y mejorando la visualización de los *rankings* de resultados.

4.2. Modelos de Clasificación Implementados

El sistema implementa tres modelos de recuperación de información clave: Jaccard (binario), Similitud de Coseno (TF-IDF), y BM25. Todos los modelos reutilizan la función

de preprocesamiento definida, asegurando que la representación de los documentos y las consultas sea consistente (tokenización, minúsculas, eliminación de *stopwords* y caracteres no alfabéticos).

1. **Similitud Jaccard (Vectores Binarios):**

Aunque el modelo implementado (*ModeloBinario*) en el código se orienta a la lógica Booleana AND (requiriendo que todos los términos de la consulta estén presentes en el documento), su base es la representación vectorial binaria.

- **Representación:** Los documentos y consultas se representan como vectores binarios (clase `ModeloBinario`), donde el valor es 1 si el término está presente en el documento y 0 si está ausente. La clase `ajustarCorpus` construye una `matrizOcurrencia` donde las filas son documentos y las columnas son términos, con valores entre 0 y 1.
- **Modelo de Recuperación:**
    - La función `buscar` utiliza una lógica de intersección estricta: D es relevante para todo termino t que pertenece a Q, el término t aparece en D.
    - Se aplica la operación AND booleana (`relevanciaBooleana = relevanciaBooleana & (vectorTermino == 1)`) sobre las filas de la matriz, garantizando una coincidencia exacta de todos los términos.

Ranking: Al ser un modelo estrictamente booleano sin ponderación, la función no genera una puntuación de similitud continua (como Jaccard) sino un resultado binario (relevante o no relevante). El *ranking* se limita a devolver los primeros k documentos que cumplen con la condición booleana, sin ordenar por un grado de relevancia.

2. **Similitud de Coseno (TF-IDF):**

El Modelo Vectorial (`ModeloVectorialTfIdf`) utiliza la ponderación TF-IDF para asignar peso a cada término, lo que permite la recuperación basada en el grado de similitud entre el vector de consulta y el vector de documento.

Ponderación TF-IDF:

- **Frecuencia de Término (tf):** Se calcula en la función `calcularTf` como el conteo simple de las veces que un término aparece en un documento.
- **Frecuencia Inversa de Documento (idf):** La función `calcularIdf` usa la fórmula:

$$idf = \log(\frac{N+1}{df_t+1}) + 1$$

donde N es el número total de documentos y dft es el número de documentos que contienen el término t.
- La matriz final Dtfidf se construye mediante el producto TF*IDF y luego se normaliza a longitud unitaria mediante la norma L2 (`normalizarMatriz`) para facilitar el cálculo de la similitud.

Métrica de Similitud (Similitud del Coseno):

- La relevancia se mide mediante la Similitud del Coseno entre el vector de consulta normalizado q y los vectores de documento normalizados d.
- Dado que tanto Dtfidf como q están normalizados (norma 1), la similitud se calcula eficientemente como el producto punto (o producto matricial):

$$\text{sim}(\mathbf{q}, \mathbf{d}) = \mathbf{d} \cdot \mathbf{q}$$

Ranking: La función `buscar` ordena los documentos de forma descendente (`np.argsort(...)[::-1]`) según el valor de la similitud de coseno, mostrando un *ranking* de los documentos más relevantes.

BM25 (Best Match 25):

El modelo BM25 (`ModeloBM25`) es un modelo probabilístico que se considera uno de los *benchmarks* más efectivos en IR, introduciendo saturación de frecuencia de término y normalización por longitud de documento.

- **Parámetros Clave:** Se utilizan los valores por defecto estándar: k1 = 1.2 (controla la saturación del tf) y b = 0.75 (controla el impacto de la longitud del documento).
- **Cálculo de IDF (BM25):** Se utiliza la versión más moderna y robusta de IDF:

$$\text{IDF}(t) = \log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)$$

- **Fórmula de Ranking:** La función `buscar` calcula la puntuación final de un documento D para una consulta Q mediante la suma de las contribuciones de cada término ti que pertenece a Q:

$$\text{Score}(D, Q) = \sum_{t_i \in Q} \text{IDF}(t_i) \cdot \frac{f(t_i, D) \cdot (k_1 + 1)}{f(t_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

donde f(ti, D) es la frecuencia del término en el documento, |D| es la longitud del documento, y avgdl es la longitud promedio de los documentos.

Ranking: La función acumula estas puntuaciones en un vector (`puntuaciones`) y ordena los documentos de forma descendente, proporcionando un *ranking* altamente calibrado por la relevancia probabilística.

Funcionalidades Transversales

- **Ejecución de Consultas de Texto Libre:** Implementado en la función buscar(consulta, k=3) de cada clase, permitiendo al usuario ingresar cualquier cadena de texto.
- **Ranking de Documentos:** Todos los modelos generan un ranking de resultados ordenados por la puntuación de relevancia calculada (Similitud del Coseno para TF-IDF, Puntuación Score para BM25).
- **Visualización de Resultados:** Aunque los métodos buscar retornan los IDs y scores, la presentación final al usuario se maneja a través del módulo CLI basado en Textual (detallado en el punto 4.3).

4.3. Interfaz Básica (CLI)

- **Descripción de la Interfaz:** Breve explicación de cómo se invoca y usa el sistema (ej. `python ir_system.py --query "consulta de ejemplo"`).
- **Funcionalidades:** Confirmación de la implementación de la ejecución de consultas y la visualización de resultados.

5. Evaluación de Resultados

5.1. Conjunto de Evaluación

- **Qrels:** Los qrels se encuentran en nuestra base de datos; mientras que la columna *“Answer”* actuaba como la base de nuestros documentos, la columna *“Question”* es, en realidad, el repositorio de qrels. *“Question”* contiene preguntas que son solventadas por la columna de documentos, y en el código de preprocesamiento localizado en Google Colab se explica su uso.

  De manera resumida, se encuentran preguntas repetidas que apuntan a documentos que la responden. Por lo tanto, nuestra tarea es buscar las preguntas más repetidas y usarlas como qrels. Aquí se presentan las preguntas más repetidas encontradas en el corpus:

  What causes Causes of Diabetes
  What is (are) High Blood Cholesterol

What is (are) Medicare and Continuing Care
What is (are) Kidney Failure: Eat Right to Feel Right on Hemodialysis
What are the treatments for Breast Cancer
What is (are) Skin Cancer
What is (are) Breast Cancer
What is (are) Colorectal Cancer
What are the treatments for Prostate Cancer
What is (are) Nutrition for Advanced Chronic Kidney Disease in Adults
What is (are) Stroke
Who is at risk for Prostate Cancer
What is (are) Leukemia
Who is at risk for Breast Cancer
What is (are) Parkinson's Disease
What is (are) Age-related Macular Degeneration
What is (are) High Blood Pressure
What is (are) Prostate Cancer

En el archivo de Google Colab, detallamos los documentos pertenecientes a cada pregunta en listas numéricas. Estos son los documentos relevantes para nuestras preguntas y se compararán con los documentos recuperados por los tres modelos.

- **Métricas a Nivel de Consulta:**
  - **Precision:** Es una proporción de los documentos relevantes recuperados por documentos recuperados.

$$Accuracy = \frac{Docs\ relevantes\ recuperados}{Docs\ recuperados}$$

  - **Recall:** Es una proporción de los documentos relevantes recuperados por documentos relevantes totales.

$$Recall = \frac{Docs\ relevantes\ recuperados}{Docs\ relevantes\ totales}$$

5.2. Análisis de Métricas

- **Métrica Global (MAP):**

  La media de las precisiones promedio de todas las consultas. Este es el indicador principal del rendimiento general del sistema.

Modelo	**Precision Promedio**	**Recall Promedio**	**MAP**
Jaccard (Binario)	0	0	0
Coseno (TF-IDF)	0.2944	0.2689	0.1801

BM25	0.3444	0.3338	0.2220

- BM25 maneja mejor la longitud del documento y la saturación de términos, lo que mejora la precisión y las otras medidas con respecto a los otros modelos. Sin embargo, TF-IDF presenta un rendimiento intermedio por su sistema de relevancia de términos en las consultas. El modelo binario no logró recuperar documentos.

6. Conclusiones

- El diseño de los tres sistemas influyó en el corpus seleccionado para que las consultas sean eficaces y acertadas.
- Se incluyó una forma de recortar las consultas, incluso en el modelo binario, para recuperar el número de documentos que se necesiten.
- Los tres sistemas de consulta realizan una indexación eficiente para disminuir el tiempo de ejecución, facilitando el uso de consultas.
- Se añadieron las métricas de evaluación correspondientes para cada modelo en consultas individuales y en promedio por todos los qrels, permitiendo identificar al mejor modelo de recuperación de información.