CREACIÓN DE UN SISTEMA DE RECUPERACIÓN DE INFORMACIÓN

Daniel Alonso Muñoz

5 DE ENERO DE 2026
UNIVERSIDAD DE GRANADA

1. Introducción.

Como hemos visto a lo largo del curso la recuperación de información es un apartado dentro del sector de la tecnología de la información es una disciplina encargada de facilitar el acceso a grandes colecciones de documentos. Más en concreto, donde el volumen de información digital crece exponencialmente, es imprescindible disponer de sistemas y herramientas capaces de indexar, organizar y recuperar documentos relevantes en base a las consultas que realicen los usuario.

En esta última práctica de la asignatura, tenemos como objetivo desarrollar un sistema básico de recuperación de información que permita al usuario hacer búsquedas textuales sobre un corpus documental. El sistema que se ha implementado abarca las fases fundamentales de un motor de búsqueda:

- Recopilación de documentos.
- Procesamiento lingüístico.
- Indexación.
- Cálculo de relevancia mediante un modelo vectorial.
- Presentación de resultados a través de una interfaz web.

Para estas tareas he creado una arquitectura cliente-servidor basada en un backend desarrollado con FastAPI y un frontend implementado con React. El sistema permite hacer consultas en lenguaje natural y devuelve al usuario un conjunto de documentos ordenados por relevancia, incluyendo además fragmentos textuales representativos que faciliten al usuario la interpretación de los resultados.

Aquí está el enlace al repositorio de GitHub:

https://github.com/daniam2004/P4-RI.git

2. Proceso completo del Sistema de Recuperación de Información.

A continuación, veremos los pasos que se llevan a cabo en el proceso completo que sigue el Sistema de Recuperación desarrollado. Veremos desde la recopilación de la colección documental hasta la obtención de los resultados para una consulta que realiza el usuario. El sistema en cuestión implementa un modelo clásico de recuperación basado en el esquema vectorial y el cálculo de similitud mediante TF-IDF y similitud del coseno.

2.1. Recopilación de la colección documental.

La colección de documento usada se compone por documentos en español procedentes de fuentes abiertas. Más en concreto, tenemos:

- Libros de dominio público obtenidos desde Project Gutenberg.
- Artículos enciclopédicos procedentes de Wikipedia en español, descargados mediante crawling directo en formato HTML y posteriormente procesados para extraer su contenido textual.

Estos documentos se descargan automáticamente desde el backend y lo almacena en una carpeta llamada "docs", construyendo de esta manera el corpus base del sistema. Cada documento se puede identificar internamente mediante un identificador numérico y se conserva su nombre de archivo original para poder mostrarlo posteriormente en los resultados de búsqueda.

2.2. Procesamiento lingüístico de los documentos.

Antes de que el sistema realice la indexación, los documentos que conforman el corpus deben pasar una serie de etapas de procesamiento lingüístico para que se pueda normalizar el texto y reducir el ruido. Este proceso se realiza de manera homogénea tanto para los documentos como para las consultas que hacen los usuario. Las fases de este proceso son las siguientes:

- Normalización léxica. Consiste en que el texto se transforma por completo a minúscula y se eliminan todos los caracteres no alfabéticos. Se preservan únicamente letras, números y espacios. Además, se hace una normalización Unicode para evitar inconsistencias debidas a codificaciones distintas.
- Tokenización. El texto se divide en tokens, es decir, se divide el texto por palabras, usando el espacio como separador principal.
- Eliminación de palabras vacías. Las palabras vacías se corresponden con aquellas palabras que no aportan significado al texto de forma individual, se tratan de artículos, preposiciones, conjunciones… Para hacer esto se ha hecho uso de la

lista proporcionada por la biblioteca NLTK, Esto nos permite reducir términos con poco valor semántico al texto.

- Filtrado de tokens poco significativos. Descartamos los tokens de longitud muy corta, ya que suelen carecer de significado relevante en el contexto de recuperación de información.
- Stemming. Por último, el proceso de stemming mediante Snowball se aplica para reducir las palabras a su raíz morfológica, permitiendo agrupar variantes de una misma palabra bajo un único término.

Una vez terminan estos pasos explicados, el resultado final es una representación limpia y normalizada de cada documento en forma de lista de tokens significativos.

2.3. Construcción del vocabulario global.

Cuando ya hemos procesado todos los documentos, se construye un vocabulario global, que tiene el conjunto de todos los términos distintos presentes en el corpus tras el procesamiento lingüístico.

Este vocabulario construye la base del modelo vectorial y determina la dimensionalidad de los vectores usados tanto por los documentos como por las consultas.

2.4. Indexación del corpus mediante TF-IDF.

En el proceso de indexación del corpus he implementado el modelo de ponderación TF-IDF.

Lo primero es que, se calcula el valor IDF global de cada término del vocabulario, teniendo en cuenta el número total de documentos del corpus y en cuántos de ellos aparece cada término.

El siguiente consiste en que, para cada documento se calcula su representación TF-IDF, obteniendo un vector numérico cuya dimensión coincide con el tamaño del vocabulario. Cada componente del vector representa la importancia de un término en concreto dentro del documento y en el conjunto del corpus.

Estos vectores se almacenan en memoria y constituyen el índice global del sistema, que será usado más adelante para dar respuesta a las consultas planteadas por los usuarios.

2.5. Procesamiento de la consulta.

Desde el frontend cuando un usuario introduce una consulta, esta sigue el mismo proceso lingüístico que los documentos del corpus:

- Normalización léxica.
- Tokenización.
- Eliminación de stopwords.
- Filtrado de tokens poco significativos.
- Stemming.

Una vez se procesa la consulta, esta se transforma en un vector TF-IDF usando el mismo vocabulario y mismos valores IDF calculados para el corpus. De esta forma la consulta y los documentos se representan en el mismo espacio vectorial.

2.6. Cálculo de similitud y recuperación de resultados.

Dentro de esta sección veremos que, para determinar la relevancia de cada documento respecto a la consulta, se calcula la similitud del coseno entre el vector de la consulta y el vector de cada documento del corpus.

Los documentos con una similitud mayor que cero se consideran candidatos relevantes. Estos resultados se ordenan de forma decreciente en función del valor de similitud y se limita el número de resultados que devuelve el buscador a un valor configurable Top-K, definido por el usuario desde la interfaz gráfica.

Además, las puntuaciones de similitud son normalizadas para facilitar al usuario la interpretación de dichas puntuaciones.

2.7. Extracción de fragmentos relevantes (snippets).

Para mejorar la interpretabilidad de los resultados que se le muestran al usuario, el sistema extrae un fragmento relevante del texto de cada documento recuperado, para poder hacer esto:

- Se eliminan cabeceras no informativas, como las introducciones de los documentos descargados de Project Gutenberg.
- El texto del documento se divide en párrafos reales.
- Se busca el primer párrafo que contenga una coincidencia literal con la frase de la consulta introducida por el usuario.

Este fragmento de los documentos se devuelve junto con el resultado y se muestra al usuario en el frontend, donde los término de consultas se resalta visualmente para facilitar su identificación.

3. Arquitectura del sistema.

El sistema de recuperación de información que he desarrollado en la práctica como bien he dicho al comienzo, sigue una arquitectura cliente-servidor, separando las responsabilidades entre la interfaz de usuario y el backend, que cuenta con los procesos de procesamiento, indexación y recuperación. Esta separación permite tener una mayor modularidad, escalabilidad y facilidad de mantenimiento.

La arquitectura general la podemos dividir en 3 bloques principales:

- Frontend.
- Backend.
- Coleccción de documentos.

Estos tres bloques interactúan entre sí mediante una API REST.

3.1. Visión general de la arquitectura.

Como idea general del sistema se puede ver que está compuesto por los siguientes elementos:

- Frontend (React). Nos proporciona la interfaz gráfica con la que interactúa el usuario.

- Backend (FastAPI). Sirve para implementar toda la lógica del Sistema de Recuperación de información.
- Colección de documentos. Son el conjunto de documentos textuales almacenados localmente y usados como corpus del sistema.

La comunicación que se produce entre frontend y backend se hace mediante peticiones HTTP en formato JSON, siguiendo el estilo REST.

3.2. Backend: FastAPI.

Para implementar el backend se ha hecho uso del lenguaje de programación Python usando el framework FastAPI, debido a su eficiencia, simplicidad y soporte nativo para APIs REST.

El backend lo hemos organizado de forma modular, separando de forma clara las diferentes responsabilidades del sistema.

3.2.1. Módulo de crawling y carga de documentos.

Esta sección es la encargada de recopilar y cargar la colección documental. Incluimos funcionalidades como:

- Descargar documentos procedentes de Project Gutenberg.
- Descargar artículos de Wikipedia en español.
- Leer documentos en distintos formatos.
- Asignar un identificador único y un nombre a cada documento.

Todos los documentos descargados son almacenados dentro de la carpeta docs. Y posteriormente en memoria para su procesamiento.

3.2.2. Módulo de procesamiento lingüístico.

El módulo de procesamiento lingüístico implementa el preprocesamiento del texto tanto para los documentos como consultas. Y cuenta con las siguientes operaciones:

- Normalización del texto.
- Tokenización.
- Eliminación de palabras vacías.
- Filtrado de tokens poco significativos.

- Stemming en español.

Este módulo tiene el objetivo de reducir la variabilidad lingüística y mejorar la calidad de la representación textual usada en la indexación y recuperación.

3.2.3. Módulo de indexación.

El tercero de los módulos que vamos a ver se corresponde con la parte de indexación, es decir, es responsable de construir el índice global del corpus. Y, sus principales funciones son:

- Construcción del vocabulario global del corpus.
- Cálculo de los valores IDF de cada término.
- Cálculo de las representaciones TF-IDF de cada documento.
- Vectorización de los documentos en el espacio del vocabulario.

El índice resultante se mantiene en memoria para permitir búsquedas eficientes sin la necesidad de recalcularlo en cada consulta.

3.2.4. Módulo de recuperación y ranking.

Como su nombre indica, este módulo se encargará de gestionar las consultas realizadas por los usuarios. Entre algunas de sus responsabilidades encontramos:

- Procesar la consulta del usuario siguiendo el mismo flujo lingüístico que los documentos.
- Transformar la consulta en un vector TF-IDF.
- Calcular la similitud coseno entre la consulta y cada documento del corpus.
- Filtrar documentos con similitud mayor que cero.
- Ordenar los resultados por relevancia.
- Limitar el número de resultados devueltos, con una variable Top-K configurable por el usuario desde el frontend.

Además de estas funciones vistas, este módulo tiene la capacidad de extraer fragmentos relevantes que contienen coincidencias literales con la consulta.

3.2.5. API REST

El backend expone algunos endpoints REST que permiten:

- Ejecutar búsquedas sobre el corpus.
- Descargar nuevos documentos.
- Reconstruir el índice global.
- Consultar el estado del sistema

Estas endpoints son el punto de comunicación entre el frontend y el sistema de recuperación.

3.3. Frontend: React.

En la sección del frontend para desarrollarla he hecho uso de React, proporcionando una interfaz web sencilla e intuitiva para el usuario.

Desde la interfaz creada el usuario puede:

- Introducir una consulta en lenguaje natural.
- Configurar el número máximo de resultados que va a visualizar.
- Visualizar los resultados ordenados por relevancia.
- Consultar el nombre del documento, la puntuación de similitud y un fragmento relevante del texto.
- Resalta términos de la consulta dentro del fragmento recuperado.

El frontend como tal, envía las consultas al backend mediante peticiones HTTP y presenta los resultados devueltos de forma estructurada y visualmente clara.

3.4. Flujo de funcionamiento del sistema.

El flujo de funcionamiento completo del sistema sigue los siguientes pasos:

- El usuario introduce una consulta desde el frontend.
- El frontend envía la consulta al backend mediante una petición REST.
- El backend procesa la consulta y calcula su representación vectorial.
- Se calcula la similitud entre la consulta y los documentos indexados,
- Se seleccionan y ordenan los documentos más relevantes.
- Se extraen fragmentos relevantes de los documentos.

- El backend devuelve los resultados al frontend.
- El frontend muestra los resultados al usuario con la información especificada en el subapartado anterior.

4. Visualización del frontend.

A continuación, veremos capturas de pantallas correspondiente al frontend desarrollado en React para esta práctica. Esto se hace con la finalidad de mostrar el funcionamiento real del sistema desde el punto de vista del usuario final para fundamentar las funcionalidades descritas en los apartados anteriores de la memoria.

Como bien hemos explicado antes, el frontend se comporta como la interfaz de acceso al Sistema de Recuperación de información, permitiendo introducir consultas, configurar algún parámetro de búsqueda y la visualización de los resultados recuperados.

4.1. Vista principal del sistema.

En esta primera captura se ve la vista principal de la aplicación web. En esta como se puede observar tenemos:

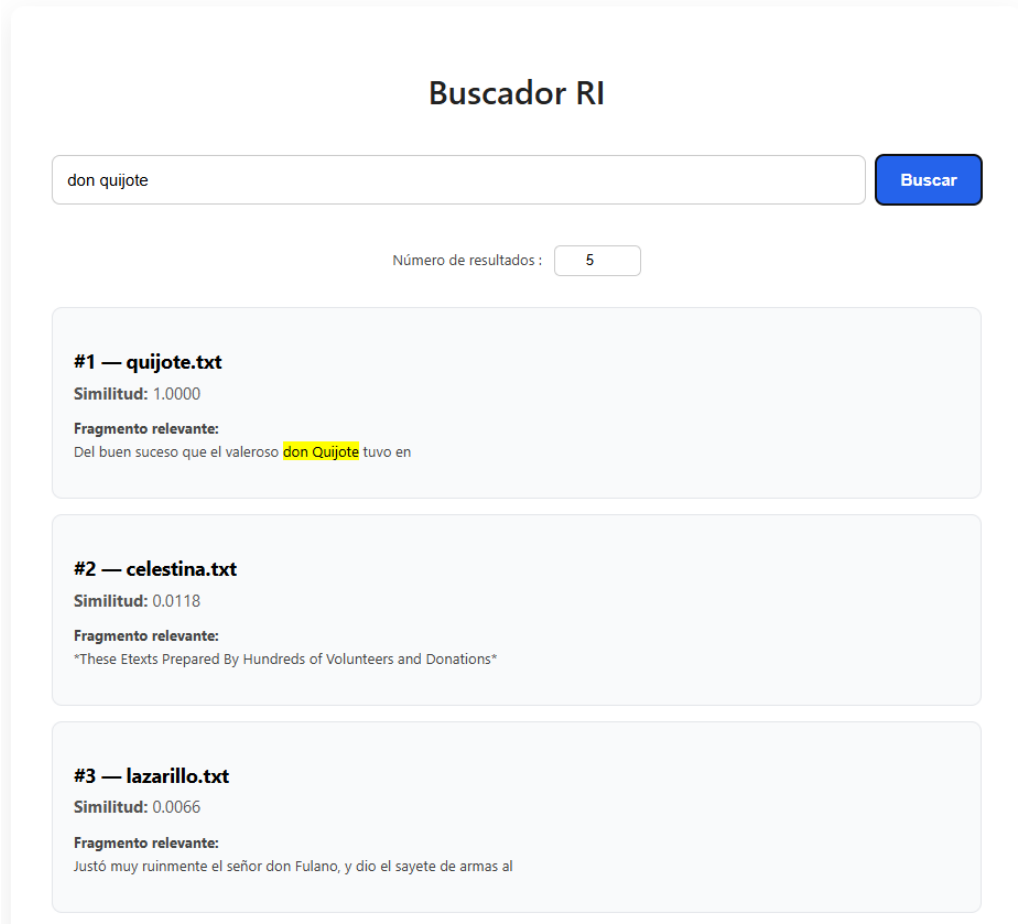
- El título del sistema.
- Un campo de texto donde el usuario introduce la consulta.
- El botón de búsqueda.
- El selector para configurar el número de resultados máximos que desea el usuario para la consulta que realice.

Es la puerta de entrada al sistema y permite como se puede observar hacer consultas de forma sencilla e intuitiva.

4.2. Introducción de la consulta.

En esta captura podemos ver como se permite introducir una consulta en el campo de texto correspondiente en lenguaje natural. Además, tras escribirla, al pulsar enter o darle al botón de buscar el sistema la devolverá los documentos más relevantes para dicha consulta.

4.3. Visualización de resultados obtenidos por relevancia.



Como se puede apreciar en la captura de pantalla, al realizar la búsqueda se le muestran al usuario los resultados correspondientes más adecuados a la consulta que ha hecho. Al ejecutar la búsqueda vemos que se muestra:

- La posición del resultado en el ranking.
- El nombre del documento.
- La puntuación de similitud respecto a la consulta.
- Un fragmento relevante del texto del documento.

En este paso los resultados aparecen ordenados de forma decreciente según su similitud con la consulta, mostrando primero los documentos más relevantes como indican las puntuaciones.

4.4. Fragmentos relevantes y resaltado de términos.

En la captura anterior vemos cómo el sistema muestra un fragmento del documento que tiene una coincidencia relevante con la consulta introducida. Los términos de la consulta que coinciden literalmente con los del fragmento del texto salen resaltados. Esto facilita

que el usuario pueda identificar rápidamente por qué ese documento ha sido recuperado y en qué contexto aparece la información que el sistema ha determinado como relevante.

Esta funcionalidad produce una mejora notable en la experiencia que puede vivir el usuario y acerca el comportamiento del sistema a los motores de búsqueda reales que se usan en los buscadores más comunes de la actualidad.

5. Conclusiones de la práctica.

Realizando esta práctica he podido desarrollar un sistema completo de Recuperación de Información que integra de forma coherente los principales conceptos teóricos explicados a lo largo de toda la asignatura. Mediante la implementación ha sido sencillo comprender de forma práctica cómo construir un motor de búsqueda básico, desde la recopilación de documentos hasta la presentación final de los resultados al usuario.

Una de las enseñanzas de las que más me he dado cuenta de su importancia es el procesamiento lingüístico de los documentos antes de la indexación. Etapas como eliminar las palabras vacías, filtrar tokens poco significativos o el stemming son básicos a la hora de querer reducir el ruido del texto y así mejorar la calidad con la que se recupera información. Aplicar el mismo proceso en documentos y consulta hace que ambos sean representados de la misma forma dentro del modelo vectorial.

Además, esta práctica me ha permitido profundizar en entender cómo funciona el modelo de espacio vectorial, el cálculo de pesos mediante TF-IDF y el uso de la similitud del coseno como medida de relevancia. Estos mecanismos, aunque conceptualmente son sencillos, he podido ver en primera persona como son efectivos para ordenar documentos según el grado de relación que tienen con una consulta, incluso en colecciones documentales heterogéneas.

Otro apartado a destacar del sistema desarrollado es la incorporación de fragmentos relevantes del texto con coincidencias literales respecto a la consulta del usuario. Poder implementar esto produce una enorme mejora en la experiencia del usuario, ya que le permite entender de forma sencilla cómo ese documento que se le muestra está correlacionado con la consulta que el usuario ha planteado.

En conclusión, esta práctica me ha permitido consolidar los conocimientos vistos a lo largo de toda la asignatura mediante una aplicación real en un sistema funcional de recuperación de información. El trabajo realizado sienta una base sólida para comprender

sistemas de búsqueda más complejos y refleja de forma clara los principios fundamentales por los que se rigen los motores de búsqueda actuales.