Nicolas Rozo Fajardo - 202112920
Nicolas Bedoya Figueroa - 202212100

INFORME DE IMPLEMENTACIÓN – TAREA 1

En este informe se presenta un resumen de las implementaciones realizadas en los notebooks, cuyo propósito fue explorar y aplicar diferentes técnicas de recuperación de información, así como evaluar su rendimiento mediante métricas clásicas del área. Cada notebook se enfoca en un componente específico del proceso, que abarca desde la definición de las métricas de evaluación hasta la implementación de dos enfoques distintos de recuperación de información. En uno de estos casos, se contrasta la implementación propia con la realizada a través de la librería GESIM. Finalmente, el informe incluye las respuestas a todas las preguntas planteadas en el enunciado de la actividad.

1. MÉTRICAS DE EVALUACIÓN PARA RECUPERACIÓN DE INFORMACIÓN

Este notebook se centró en la implementación de métricas destinadas a evaluar el desempeño de los sistemas de recuperación de información desarrollados en etapas posteriores. Se incluyeron métricas clásicas como *precision* y *recall*, junto con métricas orientadas al ranking de resultados, entre ellas *Mean Average Precision (MAP)*, *Discounted Cumulative Gain (DCG)* y *Normalized Discounted Cumulative Gain (NDCG)*. Todas las métricas fueron implementadas utilizando NumPy y, en algunos casos, se desarrollaron también sus variantes restringidas a un rango *k*. Además, cada métrica cuenta con ejemplos de ejecución claros, diseñados para ilustrar los valores obtenidos en escenarios de prueba y los parámetros necesarios para su correcta aplicación. Finalmente, se tiparon rigurosamente los argumentos y retornos de las funciones mediante la librería de tipado de Python, con el fin de especificar de manera precisa los tipos de datos esperados.

2. RECUPERACIÓN POR BÚSQUEDA BINARIA USANDO ÍNDICE INVERTIDO

Este notebook se enfocó en la implementación de un índice invertido para realizar la recuperación de documentos mediante consultas booleanas (*AND* y *NOT*). Inicialmente, se diseñó la estrategia de procesamiento de los documentos, que consiste en concatenar el título con su contenido, *tokenizar* a nivel de palabra respetando abreviaturas, palabras con guiones internos, monedas, porcentajes, números y puntos suspensivos, y tratar los signos de puntuación como *tokens* separados. Los *tokens* resultantes se normalizan, se eliminan las palabras de parada y se aplica *stemming* para reducir el tamaño del vocabulario. Tanto las palabras de parada como la estrategia de *stemming* se implementan usando la biblioteca NLTK, guardando localmente el archivo de palabras de parada para evitar descargas recurrentes.

A continuación, se *tokenizan* todos los documentos y se construye el índice invertido utilizando listas y diccionarios de Python. Cada *token* del vocabulario genera una entrada que contiene el número de documentos en los que aparece y la lista de dichos documentos, ordenada de menor a mayor. Con el índice creado, se desarrollan los algoritmos de combinación para consultas *AND* y *NOT*, así como una función encargada de ejecutar consultas mediante una pila que controla los resultados intermedios. Para facilitar la ejecución, también se implementa una función que recibe el texto de la consulta, lo preprocesa y lo convierte en un arreglo en notación postfija (RPN).

Finalmente, se evalúan todas las consultas proporcionadas en el enunciado y los resultados se almacenan automáticamente en la carpeta de resultados, indicando para cada consulta los documentos recuperados por el sistema en el formato solicitado en el enunciado.

3. RECUPERACIÓN RANQUEADA Y VECTORIZACIÓN DE DOCUMENTOS

Este notebook se centró en la implementación de la estrategia de recuperación de documentos basada en vectorización y ranking mediante TF-IDF. Inicialmente, se reutilizaron las funciones de preprocesamiento y construcción del índice invertido, ya que estos pasos son necesarios tanto para la recuperación booleana como para la vectorial. A continuación, se diseñó la estrategia de vectorización: primero se calcula la frecuencia de cada *token* en la colección de documentos, y luego, utilizando tanto estas frecuencias como la información del índice invertido (frecuencia de documentos por *token*), se construyen los vectores TF-IDF para cada documento. Dado que los vectores TF-IDF suelen ser dispersos, cada vector se representa mediante un diccionario que solo incluye los términos presentes en el documento, optimizando así el uso de memoria y evitando sobrecargas.

Posteriormente, se definió la función de similitud coseno, que aprovecha el acceso en tiempo constante a las llaves del diccionario para calcular eficientemente el producto punto entre la ponderación de los *tokens* compartidos entre los dos vectores que se están comparando. Para evitar re calcular los vectores TF-IDF de los documentos en cada consulta, se construyó un índice donde la clave es el identificador del documento y el valor es su vector TF-IDF, lo que permite una recuperación rápida durante la ejecución de consultas.

Con estas funciones implementadas, se procesaron las consultas de prueba generando para cada una su vector TF-IDF en el mismo espacio vectorial que los documentos. Se manejaron correctamente los términos fuera del vocabulario, y los resultados se almacenaron en orden descendente de similitud en el archivo de resultados. Finalmente, utilizando los documentos relevantes clasificados por expertos, se calcularon las métricas de evaluación de cada consulta, incluyendo *MAP* para evaluar el rendimiento general del sistema.

- **¿La implementación realizada es eficiente? ¿Por qué si, por qué no?**

  En lo que respecta a la eficiencia del enfoque, el sistema es razonablemente eficiente para colecciones de tamaño moderado, ya que los vectores se representan de forma dispersa y el cálculo de la similitud coseno solo considera los *tokens* compartidos entre el documento y la consulta, lo cual reduce significativamente la complejidad frente a representaciones vectoriales densas. Sin embargo, a medida que el número de documentos o la dimensionalidad del vocabulario crecen, el enfoque puede volverse menos eficiente, porque se deben comparar todos los documentos con la consulta, resultando en un costo de tiempo lineal respecto al tamaño de la colección. Para mejorar esto, se pueden utilizar mejores técnicas de indexación, al igual que mejores representaciones vectoriales que no dependan de estructuras llaves valor, las cuales ocupan más espacio del necesario debido a las relaciones que establecen.

4. RECUPERACIÓN RANQUEADA Y VECTORIZACIÓN DE DOCUMENTOS USANDO GESIM

Este notebook se centró en la implementación de la recuperación de documentos mediante vectorización y ranking utilizando Gensim. Al igual que en la estrategia TF-IDF manual, se reutilizaron las funciones de preprocesamiento y construcción del índice invertido, asegurando que los documentos y consultas se tokenizaran y normalizaran correctamente.

A diferencia de la implementación manual, Gensim crea automáticamente un diccionario de términos a partir de la colección de documentos, asignando a cada *token* un identificador único. Posteriormente, se construyen los vectores TF-IDF para los documentos utilizando las estructuras optimizadas de Gensim,

que representan los vectores de forma dispersa y eficiente, lo que permite manejar colecciones más grandes sin sobrecargar la memoria.

El cálculo de similitud entre la consulta y los documentos se realiza mediante la clase *Similarity* de Gensim, la cual construye un índice de similitud que permite recuperar los documentos más relevantes de manera rápida. Este índice aprovecha técnicas de búsqueda optimizada y acceso eficiente a los vectores dispersos, lo que reduce el tiempo de cómputo en comparación con la implementación manual que recorre todos los documentos.

Con estas herramientas, se procesaron las consultas de prueba generando sus representaciones vectoriales en el espacio definido por el diccionario de Gensim. Los resultados se ordenaron por similitud de mayor a menor y se almacenaron en el archivo de resultados. Finalmente, utilizando los documentos relevantes clasificados por expertos, se calcularon las métricas de evaluación de cada consulta, incluyendo *MAP*, para medir el rendimiento general del sistema.