Sistemas de Recuperación de Información

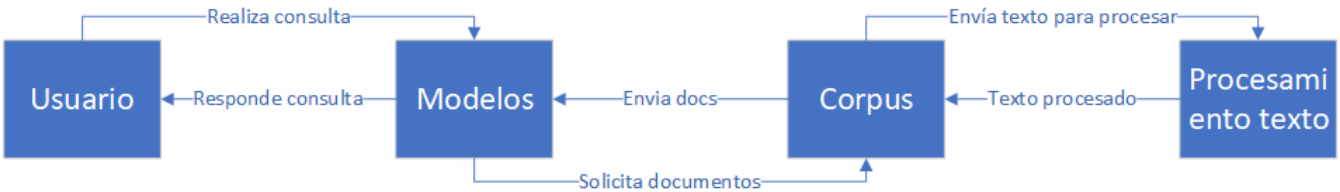
Diseño, implementación, evaluación y análisis de un Sistema de Recuperación de Información

Karlos Alejandro Alfonso Rodríguez
Karel Camilo Manresa León

Diseño e implementación del sistema

El diseño de este sistema de recuperación de información se divide principalmente en cuatro módulos:

- Corpus, documento y consulta: se encarga de todo el procesamiento de los corpus, además de la definición de documento y consulta.
- Procesamiento de texto: se encarga del análisis de texto, así como su normalización y otros procesos aplicados en texto.
- Modelos: en este módulo están implementados los modelos de recuperación de información (MRI) que se encargan de recuperar documentos a partir de una query dada.
- Usuario: este módulo comprende desde la interfaz visual, hasta la petición de búsqueda hecha por un usuario.



Procesamiento del corpus y sus elementos

Corpus

Cuando se va a procesar un corpus, el primer paso que se realiza es verificar si ya el corpus fue procesado con anterioridad, ya que como es un proceso sumamente costoso computacionalmente se realiza solo una vez.

En caso de haber sido procesado se cargan los datos anteriores generados, en caso contrario se realiza un nuevo procesamiento a dicho corpus.

Luego del procesamiento del corpus se obtienen dos diccionarios:

- `documents_words_counter : Dict[document, Counter[str, int]]` : este diccionario se encarga de mapear de documento al Counter de ese documento (`Counter` es una subclase de diccionario que se encarga se mapear `type -> int` , de modo que cuenta las ocurrencias de `type`).
- `all_words_counter : Dict[str, int]` : este diccionario se encarga de hacer un mapeo de término a cantidad de ocurrencias de ese término en el corpus.

De esta forma se obtienen la cantidad de ocurrencias de un término en un documento y la cantidad total de ocurrencias de un término en el corpus.

Documento

Para representar documentos se creó la estructura `document` que almacena distintos datos que pudieran ser importantes en un documento (título, autor, texto).

Consulta

Para el trabajo con consultas se créo la estructura `query` que almacena datos relevantes y realiza procesos necesarios sobre dicha consulta. Entre estos se encuentra texto booleano y texto limpio, más adelante se explicará su significado.

Procesamiento del texto

Para procesar texto se utilizaron diferentes herramientas, entre ellas el módulo de python NLTK (Natural Language Toolkit) para tokenizar el texto. El procesamiento del texto fue dividido en fases:

1. Tokenizar texto: es el proceso de dividir el texto en una lista de tokens, en este caso palabras, nltk proporciona una herramienta para ello.
2. Remover caracteres no ASCII: en esta fase se elimina todo caractares que no pertenezca al código ASCII.
3. Convertir a minúscula: aquí se convierten todas las palabras a letra minúscula.
4. Eliminar stopwords: esta fase es sumamente importante, en ella se eliminan todas las palabras que no proveen información útil para la clasificación del texto.
5. Eliminar puntuación: en esta fase se eliminan signos de putuación existentes en el texto (! # $ & ...)

6. Reemplazar números: aquí convertimos los números a palabras.
7. Stemming: el stemming es un proceso lingüístico que consiste en la eliminación de sufijos y prefijos de las palabras para reducirlas hasta su raíz; por ejemplo, podemos convertir consult, consultant, consulting a su raíz, que es consult.
8. Lemmatization: este proceso lingüístico persigue el mismo objetivo que el stemming aunque es un poco más comlejo, ya que utiliza también el contexto en que la palabra se está utilizando.
9. Obtención de texto booleano: esta fase se encarga de llevar un texto a su representación booleana, basándose en símbolos lógicos y de agrupamiento ('and', 'or', '( )'). Por este proceso pasan únicamente las consultas, debido a que en el SRI implementado se pueden realizar consultas booleanas.

El conjunto de procesos 1 - 6 se conoce como normalización del texto.

Modelos de recuperación de información

Al igual que el procesamiento del corpus, los modelos generan grandes cantidades de información que utilizan para ejecutarse. Nuevamente, no es necesario generar toda esta información cada vez que se utilice un modelo para recupera información, simplemente se precalcula una sola vez todos estos datos y se guardan.

Para el diseño e implementación de los modelos se utilizó el propuesto por Ricardo Baeza en el libro Modern Information Retrieval.

Modelo vectorial clásico

Para calcular la similitud entre una consulta y un documento se debe tener previamente calculada la frecuencia de los términos en los documentos (tf_{ij}), la frecuencia de documento inversa (idf_i) y los pesos de cada término en un documento (w_{ij}). En este caso se aborda de esta manera:

- tf_{ij} : Se itera por documentos y se aplica la fórmula señalada en el libro. Luego se guardan los datos en un diccionario que mapea `Dict[document, Dict[str, float]]`, donde dado un documento y un término se obtiene la frecuencia normalizada de ese término en ese documento.
- idf_i : Se aplica la fórmula planteada en el libro y se guardan los datos en un diccionario que mapea `Dict[str, float]`, donde dado un término, se obtiene un float que representa la frecuencia de ocurrencia del término dentro de todos los documentos del corpus.

- w_{ij} : Luego de aplicar la fórmula $(w_{ij} = tf_{ij} * idf_i)$, se guardan los datos en un diccionario que mapea `Dict[document, Dict[str, float]]`, de modo que dado un documento y un término, se obtiene el peso de ese término en ese documento.

A partir de los tres datos vistos anteriormente y una cosulta, se calcula la similitud entre los documentos del corpus y dicha consulta utilizando la fórmula $sim(d_j, q) = \frac{\sum_{i=1}^{n} w_{ij} * w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} * \sqrt{\sum_{j=1}^{n} w_{iq}^2}}$. Luego se establece un ranking entre los documentos recuperados.

La implementación de este modelo tiene la particularidad de que los vectores que se multiplican en la función de similitud no tienen a todos los términos del corpus, como se propone en el libro; sino que los vectores se construyen en función del tamaño de la consulta, porque de otro modo se estarían realizando muchas multiplicaciones innecesarias por cero.

Modelo booleano clásico

A diferencia de otros modelos, el modelo booleano no necesita tener datos precalculados para su ejecución, simplemente a partir de la consulta se comprueba su similitud con cada documento.

Para comprobar dicha similitud se expresó la consulta en forma normal disyuntiva (`sympy` ofrece una herramienta para ello), luego se comprobó la similitud de cada componente conjuntiva de la consulta con cada documento. Los documentos recuperados serán los que coincidan con al menos una componente conjuntiva de la consulta.

No es necesario expresarlo en forma normal conjuntiva completa como se propone en el libro, basta con hacerlo en una forma normal reducida, ya que los términos que no aparecen en una componente conjuntiva no aportan información.

Modelo fuzzy

Para calcular la similitud entre una consulta y un documento se debe tener previamente calculada la correlación entre los términos del corpus (c_{ij}), el grado de pertenencia de un documento a un conjunto difuso asociado a un término cualquiera del corpus (μ_{ij}). En este caso se aborda de esta manera:

- c_{ij} **:** Se realiza una doble iteración por todos los términos del corpus y se calcula la correlación entre cada uno, utilizando la fórmula propuesta $c_{ij} = \frac{n_{ij}}{n_i + n_j - n_{ij}}$, donde $n_i (n_j)$ es la cantidad de documentos que contienen al término $i (j)$ y n_{ij} es la cantidad de documentos que contienen ambos términos. Estos datos representan una relación entre los términos indexados con el objetivo de ampliar el conjunto de términos indexados de un documento con otros términos que se relacionen con estos, lo cual hace posible que se puedan recuperar documentos relevantes adicionales. Por ejemplo, si en una consulta

aparece el término profesor, se recuperarán seguramente documentos que contengan el término maestro.

Estos datos son almacenados en un diccionario que mapea `Dict[str, Dict[str, float]]`, donde dados dos términos se obtiene un `float` que representa la correlación entre ambos términos

- μ_{ij} : Se realiza una iteración por los documentos calculando la pertenencia de dicho documento a cada término del corpus, aplicando la fórmula propuesta $\mu_{ij} = 1 - \prod_{k_l \in d_j}(1 - c_{il})$. Estos datos son almacenados en un diccionario que mapea `Dict[document, Dict[str, float]]`, donde dados un documento y un término, se obtiene el grado de pertenencia de ese documento al conjunto difuso asociado a dicho término.

A partir de los datos anteriores y una consulta utilizando la fórmula propuesta $\mu_{qj} = 1 - \prod_{i=1}^{p}(1 - \mu_{cc_{ij}})$, se calcula la relevancia entre cada documento y dicha consulta. Donde:

- p : es la cantidad de componentes conjuntivas de la consulta.
- $\mu_{cc_{ij}}$: es el grado de pertenencia del documento d_j a la componente conjuntiva $i - ésima$ de la consulta, y se calcula:
  - $\mu_{cc_{ij}} = \prod_{l=1}^{n} t_l$, donde n es el total de términos indexados y t_l es μ_{ij} si el término l aparece positivo en cc_i, en caso contrario es $1 - \mu_{ij}$.

En el caso de este modelo se hace necesaria obtención de la forma normal disyuntiva completa para cada consulta, ya que la aparición positiva o negativa de un término en un componente conjuntiva determina información relevante.

Interfaz visual

Para la interfaz visual se utilizó el framework Streamlit, ya que permite de manera sencilla y rápida desarrollar una aplicación de datos.

Evaluación y análisis

Como parte fundamental de la creación de un SRI se tiene el proceso de evaluación y análisis, donde se determina la efectividad y calidad del mismo.

Para la evaluación de un modelo se necesitan colecciones de prueba que cuentes con:

- Una colección de documentos bien definida.
- Un conjunto de consultas a realizar sobre la colección.

- Un conjunto de evaluaciones de relevancia dada una consulta.

Las colecciones de prueba utilizadas fueron:

- Cranfield
- CISI
- Medline

Métricas utilizadas

Para realizar un análisi de la efectividad de cada modelo se emplearon distintas métricas. Primero, se define como:

- RR : Documentos relevantes recuperados
- RI : Documentos irrelevantes recuperados
- NR : Documentos relevantes no recuperados
- NI : Documentos irrelevantes no recuperados
- P : Precisión
- R : Recobrado

Luego, las métricas utilizadas son:

- Precisión:

  $$P = \frac{|RR|}{|RR \cup RI|}$$

  Denota la proporción de los documentos relevantes recuperados sobre el total de documentos recuperados.

- Recobrado:

  $$R = \frac{|RR|}{|RR \cup NR|}$$

  Denota la proporción de los documentos relevantes recuperados sobre el total de documentos relevantes.

- Medida F1:

  $$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Medida que armoniza entre Precisión y Recobrado.

- Fallout:

  $$Fallout = \frac{|RI|}{|RI \cup NI|}$$

  Denota la cantidad de documentos irrelevantes recuperados sobre el total de docuemntos irrelevantes.

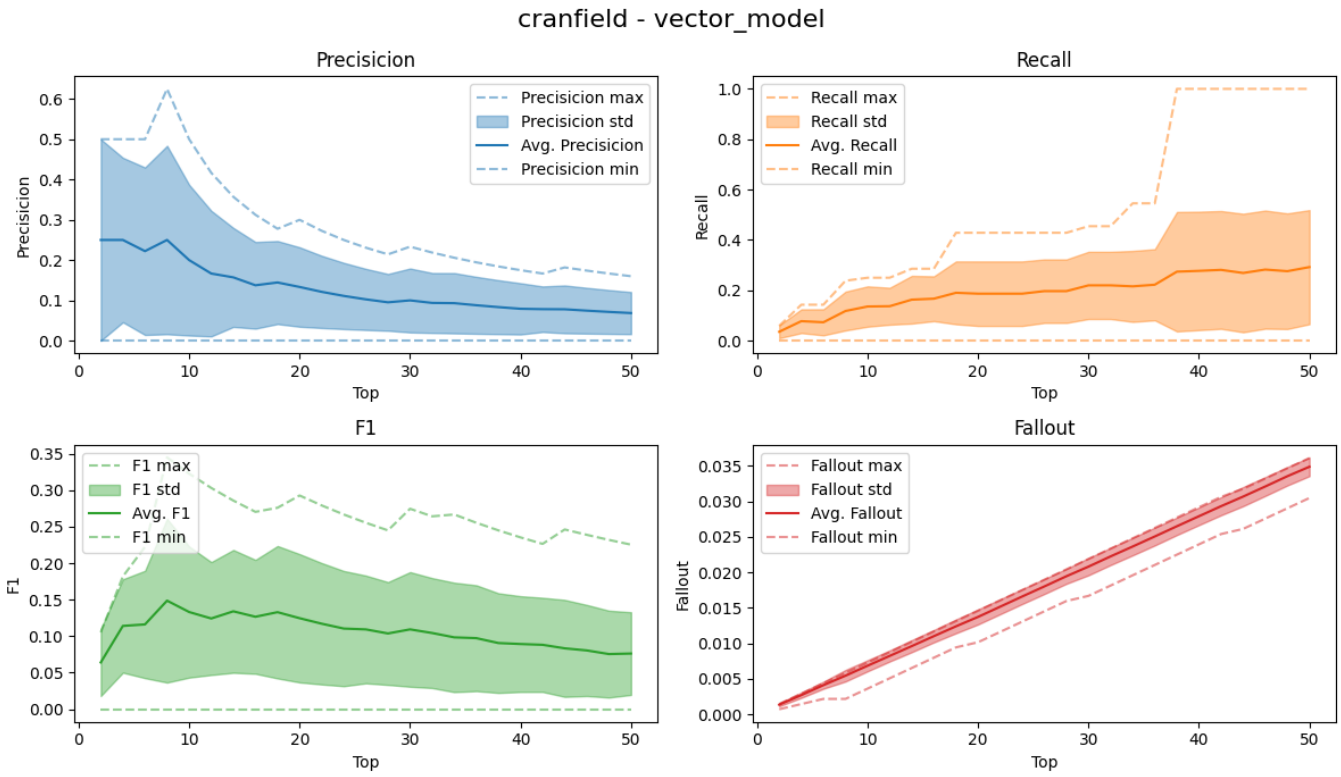
Proceso de evaluación

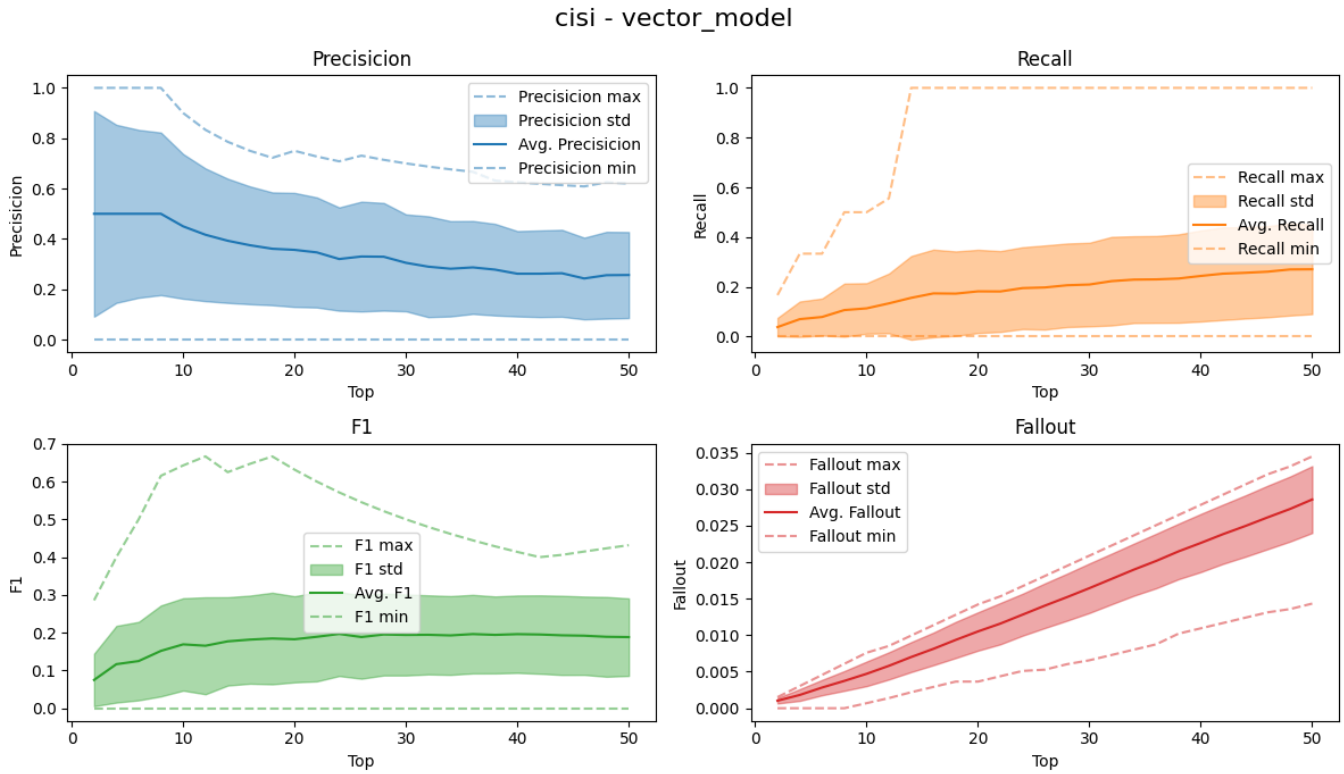
Para describir el proceso de evaluación primero es necesario definir el concepto de tope. Este concepto se define como: los k primeros documentos resultantes de una consulta.

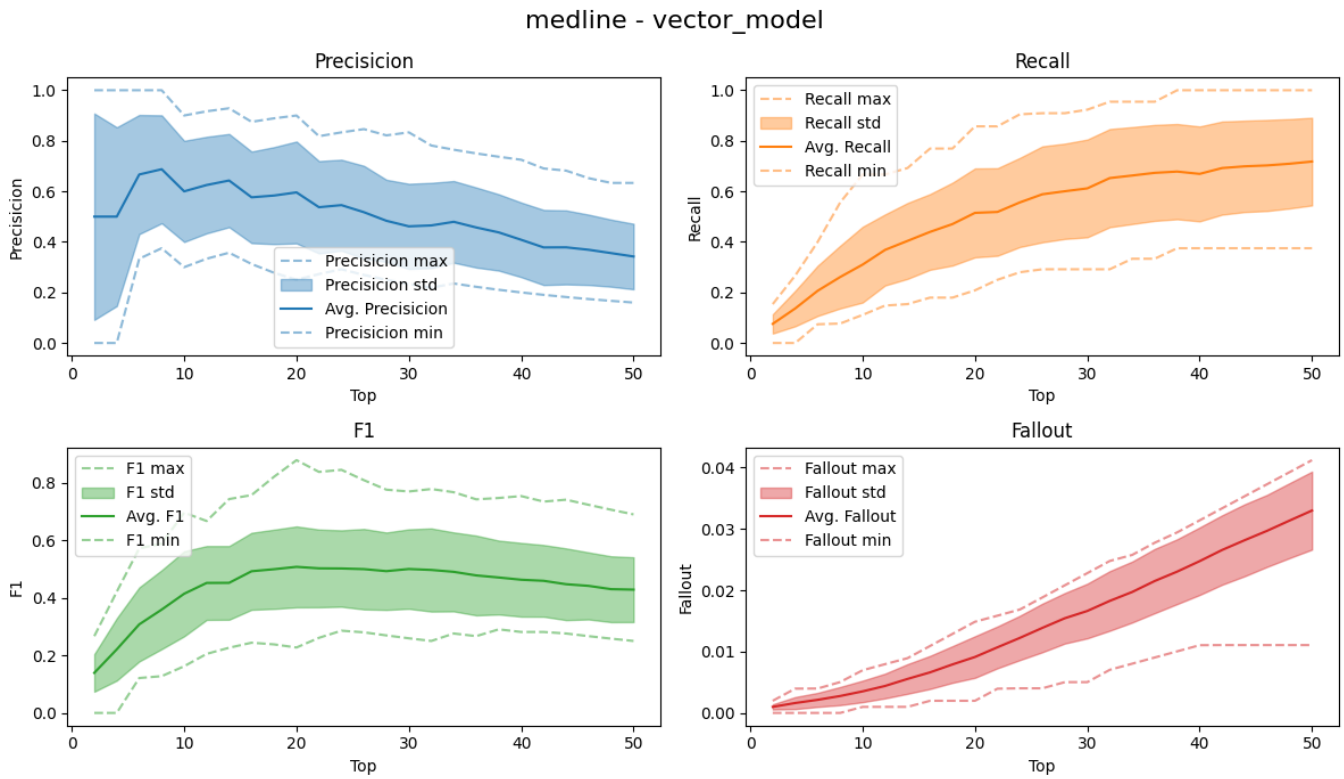
En el análisis de los modelos se calculan las métricas antes mencionadas, por cada consulta, para distintos topes. De estas métricas se extraen valores estadísticos relevantes como: media, desviación estándar, máximo y mínimo, por cada uno de los topes establecidos $(2, 4, 6, ..., 50)$.

De esta forma para cada modelo se puede analizar el comportamiento de las métricas estudiadas a medida que aumenta la cantidad de documentos recuperados.

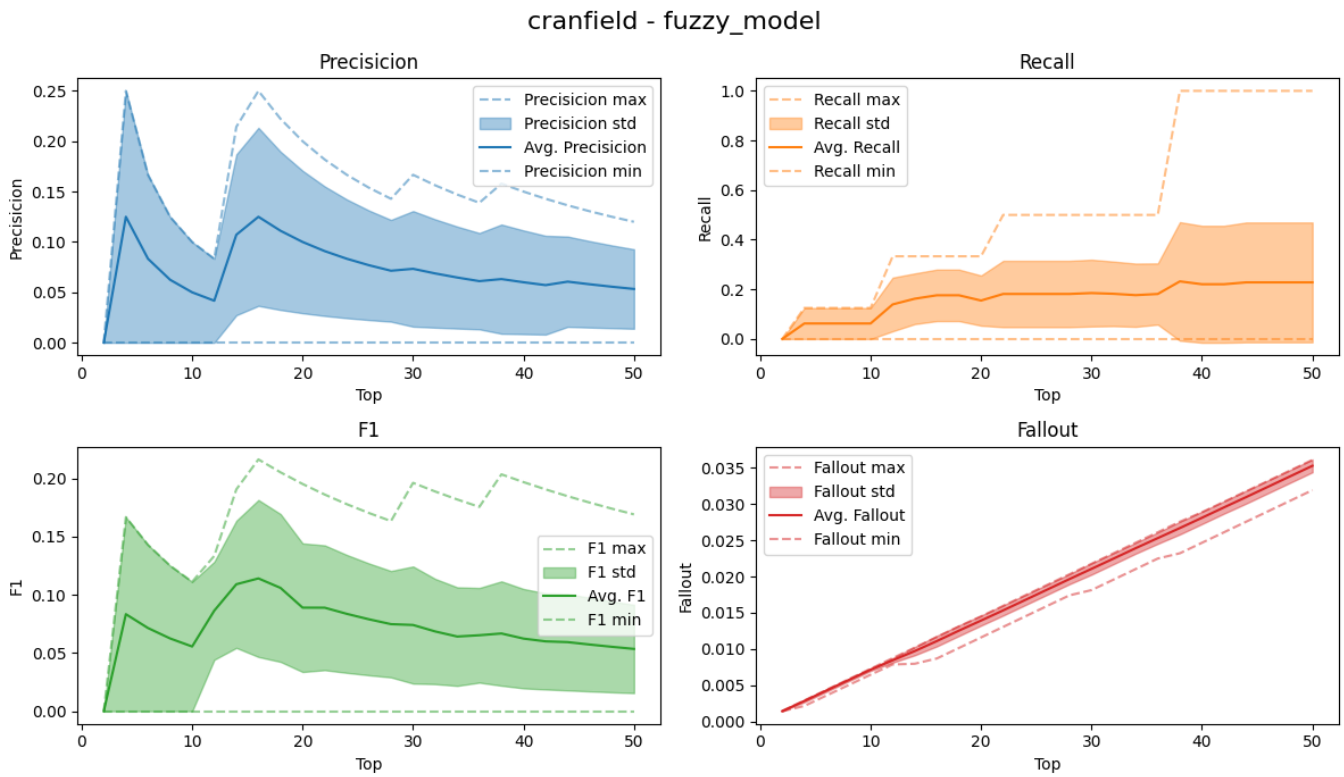
A continuación se muestran los resultados obtenidos para el modelo vectorial utilizando las colecciones: Cranfield, CISI y Medline.



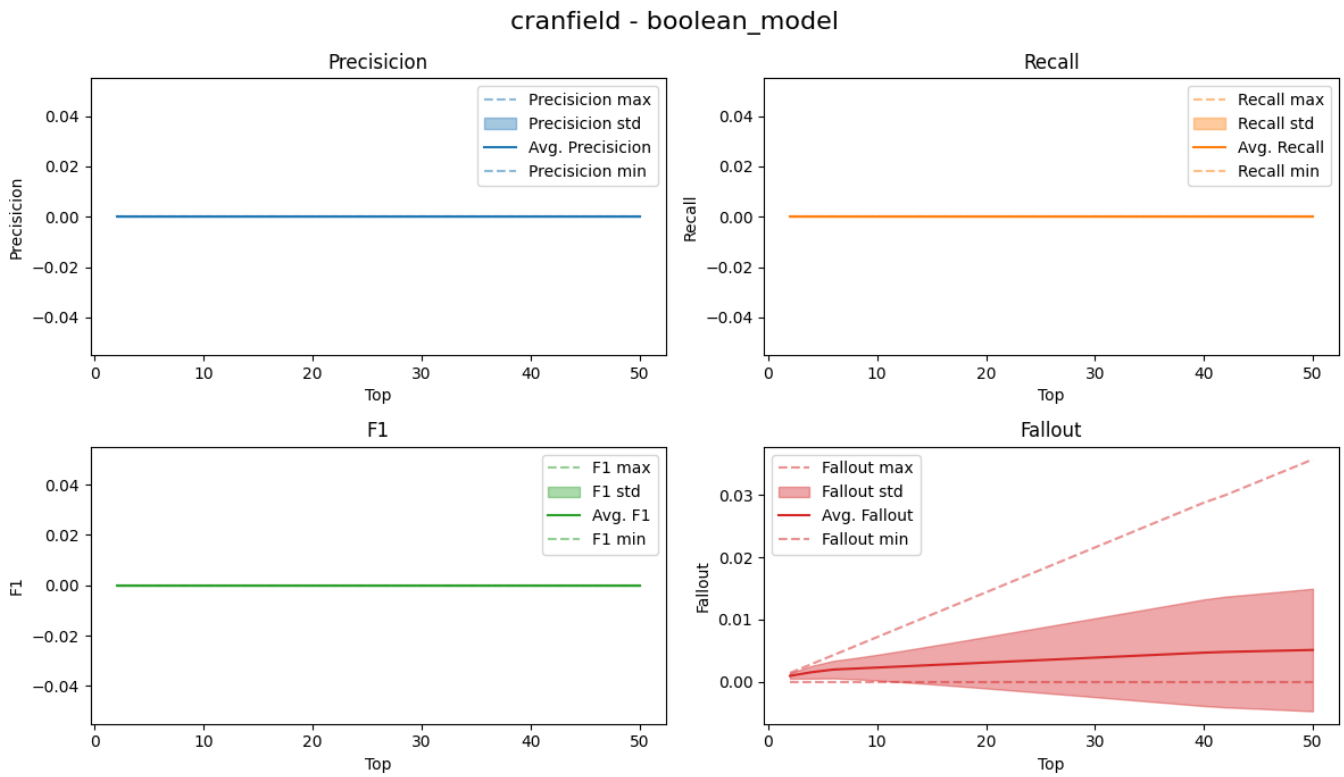




Luego estos son los resultados obtenidos de la evaluación del modelo fuzzy en la colección Cranfield:



El resultado obtenido en la evaluación del modelo booleano resulta muy curioso, ya que presenta 0 precisión y 0 recobrado. Esto es debido a que en el modelo booleano se asumen las consultas en texto booleano; o sea, siempre que no exista conector booleano entre dos términos de la consulta se interpreta como un &. Las consultas de la colección Cranfield no tienen esto en cuenta y al ser muy extensas no existe un consulta contenida completamente en algún documento.



Ventajas y desventajas

Modelo vectorial

Ventajas

Este es uno de los modelos más utilizados en la actualidad por su eficiencia y versatilidad a la hora de generar rankings de precisión en colecciones de gran tamaño. Permite además coincidencia parcial y la medida de similitud es sumamente eficaz.

Desventajas

La principal desventaja de este modelo es que al ser un modelo estadístico - matemático, no tiene en cuenta la estructura sintáctico - semántica del lenguaje natural. En general recupera documentos cuando hay igualdad de palabras entre el documento y la consulta, no tiene en cuenta, por ejemplo, sinónimos.

Modelo booleano

Ventajas

Entre las principales ventajas de este modelo está su sencilla implementación y su rápida comprensión. Además se pueden realizar consultas con precisión semántica al poder ser expresadas como booleanas.

Desventajas

La principal desventaja de este modelo es el lenguaje de la consulta, se hace complicado para usuarios no expertos realizar consultas booleanas. Además que no admite correspondencia parcial y no tiene noción de ranking.

Modelo fuzzy

Ventajas

Una de las ventajas más importantes es que este modelo tiene la capacidad de recuperar documentos que no incluyan exactamente los mismos términos de la cosulta original dada, se tiene en cuenta la relación existente entre los términos.

Una mejora significativa respecto al booleano clásico es la posibilidad de definir un ranking entre los documentos recuperados.

Desventajas

La principal desventaja es el alto costo computacional que requiere su implementación. Además no considera la frecuencia de ocurrencias de un término en un documento; es decir, la función de ranking no prioriza documentos que tengan muchas ocurrencias de un término, sobre documentos que contengan solo una vez dicho término.

Recomendaciones

Como propuestas a modo de recomendación sobre el trabajo se pueden señalar varios aspectos interesantes como: el uso e incorporación de retroalimentación a los modelos con la cual se ganaría en rendimiento y precisión del mismo; la implementación de algoritmos de Crawling para una futura integración Web; operaciones textuales más avanzadas, el uso de bases de conocimientos como ontologías, la integración de algoritmos de agrupamiento y clasificación y la incorporación de expasión de consultas.