Proyecto Final de Sistemas de Recuperación de Información

Facultad de Matemática y Computación

Universidad de la Habana

Autores:

Juan Carlos Esquivel Lamis

Juan Carlos Vázquez García

Yandy Sánchez Orosa

Carrera: Ciencias de la Computación

Resumen

La recuperación de la información (RI, Information Retrieval) es la rama de la Ciencia de la Computación que se ocupa de extraer información de documentos no estructurados (cartas, periódicos, artículos, etc.) de los cuales, a diferencia de los datos con estructura (almacenados en bases de datos) no es fácil sacar información útil automáticamente. En este trabajo se presentarán dos (2) modelos de recuperación de información para llevar a cabo una comparación de resultados obtenidos por dichos modelos en dos (2) colecciones de datos diferentes.

Abstract

Information Retrieval (IR) is the branch of Computer Science that deals with extracting information from unstructured documents (letters, newspapers, articles, etc.) from which, unlike data with structure (stored in databases) it is not easy to extract useful information automatically. In this work, two (2) information retrieval models will be presented to carry out a comparison of the results obtained by said models in two (2) different data collections.

Introducción

La RI es la parte de la ciencia de la computación que estudia la recuperación de la información (no datos) de una colección de documentos escritos. Los documentos recuperados pueden satisfacer la necesidad de información de un usuario expresada normalmente en lenguaje natural. Se puede definir la RI como la localización y presentación a un usuario de información relevante a una necesidad de información expresada como una pregunta. Un sistema de recuperación de información procesa archivos de registros y peticiones de información, e identifica y recupera de los archivos ciertos registros en respuesta a las peticiones de información. Algunos de los nombres como se les conoce a estas técnicas son: Sistema de almacenamiento y recuperación de la información, sistema de recuperación de información, sistema de recuperación de textos y base de datos de información no estructurada. Informalmente el problema lo podríamos definir como sigue: Dada una necesidad de información (consulta + perfil del usuario) y un conjunto de documentos, ordenar los documentos de mayor a menor relevancia para esa necesidad y presentar un subconjunto de los más relevantes.

Existen varios modelos de RI, que comúnmente se dividen en modelos clásicos y modelos extendidos. Decidimos utilizar los modelos booleano y vectorial (ambos pertenecientes a los modelos clásicos) para resolver los objetivos de este trabajo. Más adelante explicaremos cómo funcionan estos modelos y las razones por las que fueron escogidos.

Este informe se encuentra estructurado de la siguiente manera: en el capítulo 1 se da una breve descripción de los métodos booleano y vectorial, en el capítulo 2 explicamos cómo desarrollamos estos métodos de recuperación de información y las funcionalidades adicionales implementadas y, por último, en el capítulo 3 se realiza la comparación de los resultados obtenidos por ambos métodos, así como las ventajas y desventajas de ambos.

Objetivos del Trabajo

- ✓ Realizar un sistema donde se usarán dos (2) modelos de recuperación de información sobre dos (2) colecciones de datos.
- ✓ Diseño completo del sistema según cada etapa de la recuperación de información. Especificar las consideraciones y argumentos para la elección y aplicación de los modelos y enfoques adoptados.
- ✓ Presentación de las herramientas empleadas para la programación y aspectos más importantes del código.

- ✓ Evaluación del sistema empleando las métricas objetivas y subjetivas estudiadas en clase (incorporar consultas de ejemplo con los resultados proveídos por la solución, al menos una por cada colección de prueba).

- ✓ Análisis crítico de las ventajas y desventajas del sistema desarrollado.

- ✓ Recomendaciones para trabajos futuros que mejoren la propuesta.

Capítulo I: Modelos Booleano y Vectorial

Modelo Booleano

En el modelo booleano los documentos son vistos como un conjunto de términos y las preguntas como expresiones booleanas. Si designamos un documento como D y una pregunta como P, para cada par (documento, pregunta) se asigna un valor de similitud (número real) dado por la función $S : D \times P \rightarrow R$, donde D y P son conjuntos de términos. Por lo general un documento es representado como un registro de que contiene ceros (0) y unos (1), dependiendo de si contiene un término o no.

Este modelo se construye a partir de las reglas clásicas de la lógica de conjuntos, y sus operadores son unión, intersección y negación; además, es muy común implementar una combinación de las anteriores cláusulas.

En este modelo la relevancia es binaria: un documento es relevante o no lo es. Cuando consultamos una palabra el modelo la considera relevante si y solo si el documento la contiene; por ejemplo, en las consultas que utilizan "and" los documentos deben contener todas las palabras de la consulta; en las consultas "or" los documentos deben contener alguna palabra y en consultas "not" A los documentos no deben contener A.

Modelo Vectorial

El modelo vectorial fue definido por Gerard Salton, profesor de ciencia de la computación de la Cornell University. Este es el modelo más usado en operaciones de RI, así como también en operaciones de categorización automática, filtrado de información, etc., pues permite dar un grado de pertenencia de un documento a una pregunta, a diferencia del modelo booleano que determina sólo si el documento pertenece o no a la pregunta.

En este modelo se seleccionan las palabras útiles, que por lo general son todos los términos del texto excepto las palabras vacías; este proceso se enriquece utilizando técnicas de lematización y etiquetado.

Sea $\{t_1, t_2, \dots, t_k\}$ el conjunto de los términos y $\{d_1, d_2, \dots, d_n\}$ el conjunto de los documentos. Podemos representar un documento d_i como el vector

$$d_i \rightarrow \vec{d}_i = \{w(t_1, d_i), \cdots, w(t_k, d_i)\},$$

donde $(w(t_r, d_i)$ es el peso del término t_r en el documento d_i. Cada documento puede ser visto como un punto en el espacio vectorial con tantas dimensiones como términos tenga el diccionario de indexación.

La representación de las preguntas al igual que los documentos tiene la forma $P = \{w.p_1, w.p_2, \cdots, w.p_n\}$, donde n es el número de términos distintos en la colección.

Hay varias formas de calcular los pesos; una de las más populares es considerar:

$$w(t_r, d_i) = w_{r,i} = \frac{tf_{r,i} \times idf_r}{\overrightarrow{d_i}} = \frac{tf_{r,i} \times \log\frac{N}{n_r}}{\sqrt{\sum_{s=1}^{k}\left(tf_{s,i} \times \log\frac{N}{n_s}\right)^2}}$$

donde $tf_{r,i}$ (frecuencia del término), es la cantidad de veces que t_r aparece en d_i, y n_i es la cantidad de documentos donde aparece t_r. Si un término aparece mucho en un documento, se supone que es importante en ese documento, entonces tf crece. Pero si por otra parte el término aparece en muchos documentos, entonces no es útil para distinguir un documento de los otros, así que idf decrece. Además, podemos normalizar las frecuencias para no favorecer documentos más largos. En otras palabras, lo que intentamos medir es cuándo ayuda ese término a distinguir ese documento de los demás. Podemos calcular la similaridad entre dos documentos mediante la distancia coseno,

$$sim(d_i, d_j) = \overrightarrow{d_i} \cdot \overrightarrow{d_j} = \sum_{r=1}^{k} w_{r,i} \times w_{r,j},$$

que geométricamente corresponde al coseno del ángulo entre los dos vectores.

La similaridad es un valor entre cero y uno, así que todos los documentos iguales tienen similaridad uno (1) y los ortogonales (es decir, que no comparten términos) tienen similaridad cero (0). Una consulta puede verse como un documento formado por palabras, y por lo tanto como un vector. Dado que en general cada palabra aparece una sola vez en la consulta y esta es muy corta, se puede en general calcular la relevancia de d_i para q con la fórmula

$$sim(d_i, q) = \sum_{r=1}^{k} w_{r,i} \times w_{r,q} \sim \sum_{r=1}^{k} w_{r,i} \times idf_r .$$

El modelo más simple para calcular la similitud entre una consulta y un documento, utilizando el modelo vectorial, es realizar el producto escalar de los vectores que los representan; en esta ecuación no se incluye la normalización de los vectores, a fin de obviar las distorsiones producidas por los diferentes tamaños de los documentos

Capítulo II: Implementación

Procesamiento de Datos

Para el procesamiento de los datos utilizamos 3 técnicas de procesamiento de lenguaje natural: ***tokenization***, ***lemmatization*** y ***stemmization***. Para esto usamos los módulos de Python ***nltk*** y ***re***. La librería ***nltk*** proporciona diferentes métodos para el procesamiento de texto en lenguaje natural.

Primero eliminamos del texto los símbolos de puntuación y dígitos con el uso de la biblioteca ***re***, y dividimos el texto en tokens posteriormente. Después normalizamos el texto llevando cada token a minúsculas y eliminando las *stopwords*, que no son mas que palabras en el texto inicial que no nos aporta información relevante sobre el texto, por ejemplo, pronombres, preposiciones, conjunciones, etc. Por último, usando la clase ***SnowballStemmer*** aplicamos ***stemmization***, que es un método utilizado para reducir una palabra a su raíz o en inglés ***stem***. Estas raíces son la parte invariable de palabras relacionadas sobre todo por su forma. De cierta manera se parece a la ***lemmatization***, pero los resultados (las raíces) no tienen por qué ser palabras de un idioma. Por ejemplo, el algoritmo de *stemming* puede decidir que la raíz de *amamos* no es *am-* sino que pudiera ser *amam-*.

El ***stemming*** es mucho más rápido desde el punto de vista del procesamiento que la lematización. También tiene como ventaja que reconoce relaciones entre palabras de distinta clase.

Una desventaja del ***stemming*** es que sus algoritmos son más simples que los de lematización. Pueden "*recortar*" demasiado la raíz y encontrar relaciones que realmente no existen entre palabras (***overstemming***). También puede suceder que deje raíces demasiado extensas o específicas, y que tengamos más bien un déficit de raíces (***understemming***), en cuyo caso palabras que deberían convertirse en una misma raíz no lo hacen.

La ***tokenization***, ***lemmatization*** y ***stemmization*** de un texto suelen ser procedimientos fundamentales para muchas tareas relacionadas con la extracción automática de características y de datos de los textos. Estos procedimientos están en la base de los grandes y pequeños buscadores de información. El ***stemming*** suele ser una buena solución cuando no importa demasiado la precisión y se requiere de un procesamiento eficiente. La lematización suele funcionar mejor cuando se necesita procesar palabras de manera similar a como lo hace un ser humano. Haz pruebas de precisión y rendimiento y quédate con el procedimiento que mejor se adapte a tus textos y requerimientos.

Expansión de consultas

En el contexto de los motores de búsqueda, la expansión de consultas involucra evaluar una entrada del usuario (las palabras que el usuario ingresa en el área de consulta de búsqueda, y a veces otros tipos de datos) y expandir la consulta de búsqueda para que se ajuste a documentos adicionales.

La expansión de consultas involucra técnicas como:

- Encontrar sinónimos de palabras, y buscar también por los sinónimos.
- Encontrar las varias formas morfológicas de las palabras involucradas en la búsqueda, aplicando técnicas de lematización y ***stemming***.
- Reparar errores tipográficos y buscar automáticamente por la forma corregida o sugerirla.
- Ponderar los resultados según la relevancia.

Para la expansión de consultas usamos el paquete ***wordnet*** de la biblioteca ***nltk***, la cual nos proporciona una base de datos léxica para múltiples idiomas. Los términos similares de varios idiomas se conectan mediante ***synsets*** (conjunto de sentidos). ***Wordnet*** se puede usar para obtener términos relacionados para un término en particular en varios idiomas y puede ayudar a satisfacer la necesidad de información del usuario.

Por lo que apoyándonos en ***wordnet***, le permitimos al usuario agregar a la consulta inicial aquellas palabras que presentan mayor similitud con los términos de la consulta.

Crawling

Un web *crawler* es un servicio de Internet que ayuda a los usuarios en su navegación web mediante la automatización de la tarea de cruce de enlaces, la creación de un índice de búsqueda de la web y el cumplimiento de las consultas de los buscadores desde el índice. Es decir, un web *crawler* descubre y recopila automáticamente diferentes recursos de Internet de manera ordenada de acuerdo con los requisitos del usuario. Estos programas explotan la estructura en forma de grafo de la web para moverse de una página a otra.

En su forma más simple, un *crawler* comienza desde una página semilla y luego usa los enlaces externos dentro de ella para llegar a otras páginas. El proceso se repite con las nuevas páginas que ofrecen más enlaces externos a seguir, hasta que se identifique un número suficiente de páginas o se alcance algún objetivo de mayor nivel.

En nuestro sistema implementamos un algoritmo de *crawler* llamado **FishSearch**.

FishSearch propone un paradigma más refinado que puede explicarse mediante la siguiente metáfora. Compara a los agentes de búsqueda que exploran la Web con un banco de peces en el mar. Cuando se encuentra alimento (información relevante), los peces (agentes de búsqueda) se reproducen y continúan buscando alimento, en ausencia de alimento (sin información relevante) o cuando el agua está contaminada (ancho de banda pobre), mueren.

El principio clave del algoritmo es el siguiente: toma como entrada una URL inicial y una consulta de búsqueda, y crea dinámicamente una lista de prioridades (inicializado en la URL inicial) de las siguientes URL (en adelante, nodos) que se explorarán. En cada paso, el primer nodo se extrae de la lista y se procesa. A medida que el texto de cada documento está disponible, se analiza mediante un componente de puntuación que evalúa si es relevante o irrelevante para la consulta de búsqueda (valor 1-0) y, en función de esa puntuación, una heurística decide si proseguir la exploración en esa dirección o no. Cada vez que se recupera un documento de origen, se escanea en busca de enlaces. A los nodos a los que apuntan estos enlaces (hijos) se les asigna un valor de profundidad. Si el padre es relevante, la profundidad de los hijos se establece en algún valor predefinido. De lo contrario, la profundidad de los hijos se establece en uno menos que la profundidad del padre. Cuando la profundidad llega a cero, la dirección se elimina y ninguno de sus elementos secundarios se inserta en la lista.

Los hijos cuya profundidad es mayor que 0 se insertan en la lista de prioridad de acuerdo con las siguientes heurísticas:

- Los primeros hijos de $\alpha *$ ancho ($\alpha > 1$) de un nodo relevante se agregan al encabezado de la lista.
- Los primeros k-hijos (k = ancho) de un nodo irrelevante se agregan a la lista justo después del último hijo de un nodo relevante.
- El resto de los hijos se agregan a la cola para manipular solo si el tiempo lo permite.

Capítulo III: Evaluación de Resultados

En nuestro trabajo implementamos dos (2) modelos para la recuperación de información, el modelo booleano y el modelo vectorial. A continuación, presentaremos las ventajas y desventajas de cada uno de ellos, que nos ayudaran a explicar el porqué de nuestra elección.

Modelo Booleano

Entre las principales ventajas:

- Es un modelo simple basado en teoría de conjuntos.
- Es muy fácil de implementar y de entender.
- Las consultas tienen una necesidad semántica, aunque no siempre es fácil traducir una necesidad informacional a una expresión booleana.

Algunas de las desventajas de este enfoque son:

- No discrimina entre documentos de mayor y menor relevancia (no ofrece un ranking de documentos).
- No importa que un documento contenga una o cien veces las palabras de la consulta (Todos los términos son igual de importantes).
- Solo recupera los documentos donde la coincidencia es exacta, no hay correspondencia parcial.
- Dada la correspondencia exacta se puede considerar como un modelo de recuperación de datos, no de información.
- No considera una clase parcial de un documento (Ej: que cumpla con casi todas las cláusulas de un "Y").

Modelo Vectorial

Entre sus principales ventajas podemos apreciar:

- El esquema de ponderación $tf - idf$ para los documentos, mejora el rendimiento de la recuperación.
- La estrategia de coincidencia parcial permite la recuperación de documentos que se aproximen a los requerimientos de la consulta.
- La fórmula del coseno ordena los documentos de acuerdo al grado de similitud con la consulta, estableciendo de esta forma un ranking de acuerdo a la relevancia del mismo.

Su principal desventaja es

- Asume que los términos indexados son mutuamente independientes.

En la implementación del modelo booleano usamos ***stemming*** en los términos de la consulta y de esta forma realizamos las consultas para buscar coincidencias parciales y no exactas, y poder resolver de esta forma una de las principales desventajas del modelo booleano.

Además, se calcularon las medidas de evaluación precisión, recobrado y la medida f1, esta última un caso particular de la medida f cuando el parámetro $\beta = 1$ (se le otorga el mismo peso o énfasis a la precisión y al recobrado). Estas medidas solo se implementaron en el modelo vectorial debido a que las colecciones de datos usadas en nuestro sistema carecían de una lista de resultados para consultas en expresiones booleanas con los cuales comparar y poder evaluar nuestro modelo booleano.

Conclusiones

Consideramos que nuestro Sistema para la Recuperación de Información cumplió con los principales objetivos propuestos al inicio del mismo. Faltando algunos como la implementación de retroalimentación en los dos (2) modelos implementados, el uso de bases de conocimiento como ontologías o tesauros. Además, consideramos que se pueden mejorar aspectos del mismo, por ejemplo, en la expansión de consultas se pudiese agregar a nuestra implementación los métodos basados en ***Relevance feedback***, los cuales ejecutan la consulta inicial en el repositorio y extraen los mejores k-documentos. Luego, dos documentos obtenidos se utilizan para mejorar el rendimiento de la recuperación. Suponiendo que los documentos iniciales recuperados son relevantes y, por lo tanto, se pueden usar para extraer términos de expansión. Además, en el algoritmo de *crawler* se pudiesen mejorar algunos de los problemas que presentan los web *crawlers*, como: la canonización de URL, el análisis de páginas HTML y la ética de tratar con servidores web remotos.