Proyecto Final SRI

3Modelos-SRI

Tony Cadahía Poveda, Alejandra Monzón Peña y Leonardo Ulloa Ferrer

Facultad de Matemática y Computación, Universidad de La Habana

Resumen Este informe ofrece detalles sobre la implementación y optimización de tres modelos de recuperación de informaci: booleano, vectorial y vectorial generalizado, implementados en el lenguaje de programación C# y que utilizan Razor para la interfaz de usuario. Se presenta cómo funcioan los modelos en varias colecciones de documentos y se realiza un análisis de calidad de los modelos que contempla como medidas de evaluación la Precisión, el Recobrado y la medida F.

1. Generalidades

Los modelos que presentamos, aunque poseen características propias que hacen que sea variado su modo de funcionar, existen procesamientos comunes en todos ellos y en el proyecto en general; por ejemplo se requiere de un corpus procesado, los resultados de los modelos se evalúan con las mismas medidas de calidad y comparten una interfaz visual común. Precisamente a cada uno de estas aspectos generales se dedica una sección en el presente documento, así como para cada modelo una sección particular con detalles relevantes de su implementación.

2. Lectura y almacenamiento del corpus

La lectura de documentos se hizo posible con la implementación de un tipo para encapsular el concepto de documento, el cual lo interpreta como un iterador de caracteres. Para evitar el consumo innecesario de memoria la lectura del documento se realiza de forma perezosa (lazy). Adems, recibe una funcin para obtener metadatos del documento que va a estar en correspondencia con la coleccin de la que procede, dichos metadatos tambin se obtienen en demanda. De esto se encarga la clase Document del subproyecto DP.

Para el manejo de documentos múltiples presentes en un mismo archivo, se utilizó la misma idea de un iterador en este caso se manejan Streams para acceder eficientemente a los documentos como si se tuviera un enumerador.

Los documentos una vez procesados se almacenan en una estructura de diccionario que relaciona términos con documentos haciendo uso de un hashset, en dicha estructura para una llave (término) se tiene una LinkedList de documentos que lo contienen. Los términos presentes no incluyen palabras de paso

(StopWords) y se les representa por su abreviatura (stem) que se obtiene al aplicarle el algoritmo de Porter Stmming a cada término antes de agregarlo al diccionario. En la estructura se almacena la cantidad de repeticiones que tiene el término en el documento, información que facilita la ejecución de los modelos. De la construcción y actualización de esta estructura se encarga una clase que implementa IStorage en el subproyecto SRI

3. Interfaz de usuario

En la interfaz grfica se utiliza la tecnologa Razor Pages, en este proyecto se tienen 3 páginas con la que puede interactuar el usuario. La primera permite seleccionar los corpus que se deseen explorar. La segunda página está compuesta por un textbox para escribir la consulta y un botn para efectuarla; al efectuarla se listan los documentos en orden decreciente de la similitud con la consulta.

En este componente se unifiacan las funcionalidades do los subproyectos previamente mencionados (DP y SRI), obtiene los documentos de las direcciones especificadas, ordena el preprocesado de los mismos y cuando se efecta una consulta interacta con el ISRIModel parar obtener la lista de documentos con su respectivo ndice de similitud, los ordena en funcin de este valor y los muestra utilizando el ttulo y una breve seccin de texto obtenido de los metadatos como identificador de cada uno.

Una vez que se obtienen los resultados más relevantes a una consulta, el usuario puede leer un archivo en específico clickeando en el enlace del documento presente en el título, esta acción abre una pestaña nueva en el navegador, la cual sería la tercera página de la componente visual. En esta página se muestra el título y el texto del documento seleccionado.

4. Modelo Vectorial

Para este modelo la representación del corpus a partir de los términos indexados facilitó mucho el análisis de similitud entre los documentos y las consultas. Además del corpus procesado en un diccionario se tiene una representación del vocabulario del corpus en la que se precalcula para cada término su IDF y se interpreta cada consulta como si fuera otro documento. De esta forma se grantiza que en un solo recorrido por la estructura IStorage se tenga calculado la similitud entre la consulta y todos los documentos.

5. Modelo Booleano

Al ser necesario una forma simple de escribir una consulta booleana y de poder evaluar los vectores de los documentos para determinar la similitud, se

utilizó para ello una gramática libre del contexto que representa expresiones booleanas. Este modelo recibe consultas que son cadenas de dicha gramática y las parsea haciendo uso de un parser predictivo descendente, para cada query construye el AST(árbol de sintaxis abstracta), y asigna el valor de similitud de un documento con la consulta como el resultado de evaluar el AST con el vector documento, vector que se obtiene del corpus almacenado en el IStorage.

Para el AST se utilizó un árbol booleano donde cada nodo es una operación lógica concreta. La implementación de estas clases está en el archivo BooleanModelUtils.cs del subproyecto SRI.

6. Modelo Vectorial Generalizado

La principal característica de este modelo es su ineficiencia desde el punto de vista de programación. Su costo computacional es elevado en lo que a espacio de almacenamiento se refiere porque requiere del cómputo de los minterms que son vectores de dimensiones exponenciales y posterior a esto se tiene el cálculo que los vectores de los términos que se obtienen a partir de los minterms. No obstante se logró poner en funcionamiento este modelo guardando el valor de los vectores de los términos en archivos a medida que se iban computando.

Realizar la operación de guardado de forma eficiente resultó también un proceso complejo por lo que se utilizó el Spam de C# que permite utilizar un espacio en memoria y convertir de un tipo a otro a conveniencia en la interpretación de los datos almacenados en este espacio reservado, para haciendo provecho de estas facilidades para escribir en un archivo todo el contenido de este bloque de memoria.Realizando este tratamiento a los datos se logra crear el corpus en un tiempo promedio de 1.20 minutos para 18 000 documentos y obtener resultado para queries en 8.0 segundos.

7. Evaluación de los modelos

Para evaluar los modelos, como para las colecciones de NewsGroup y Reuters no se tienen set de consultas y relevancias, se utilizó entonces la colección de Cran y otras colecciones presentes en la librería ir-dataset de python. Como python ofrece muchas herramientas para trabajar con estas librerías, vinculamos entonces un archivo de python al proyecto para las medidas de evaluación. El código de python en este sentido funciona en dos direcciones, primeramente extrae de los sets de consultas las consultas propiamente y de los documentos su información, serializa estos datos en un JSON que se le pasa al código de C# que contiene los modelos, este ejecuta las consultas y serializa en otro JSON los resultados, con los que nuevamente interactúa python para evaluar los resultdos según precisión, recobrado y medida F.

7.1. Resultados de las Evaluaciones

8. Conclusiones

Se concretó la implementación de tres modelos de recuperación de informaci booleano, vectorial y vectorial generalizado. Ideas como utilizar .NET como Framework, debido a la rapidez de C#, la planificación de una estructura que redusca los costos de acceder al corpus, y el empleo de funcionalidades del lenguaje como el Spam, Iteradores, entre otros, permiten la optimización del funcionamiento de los modelos. Utilizar Razor para la interfaz posibilita la existencia de una forma de uso de los modelos simple para los usuarios.