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 información: booleano, vectorial y vectorial generalizado, implementados en el lenguaje de programación C# y que utilizan Razor para la interfaz de usuario. Se presenta cómo funcionan 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 F1.

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). Además, recibe una función para obtener metadatos del documento que va a estar en correspondencia con la colección de la que procede, dichos metadatos también 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 StreamReaders [1] 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 un Diccionario 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 Stemming [2] 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 gráfica se utiliza la tecnología Razor Pages, en este proyecto se tienen 3 páginas con la que puede interactuar el usuario. La primera página permite seleccionar los corpus que se deseen explorar. La segunda está compuesta por un cuadro de texto (textbox) para escribir la consulta y un botón para efectuarla; al efectuarse la consulta se listan los documentos en orden decreciente de similitud con la consulta.

En este componente se unifican las funcionalidades de los subproyectos previamente mencionados (DP y SRI), obtiene los documentos de las direcciones especificadas, ordena el preprocesado de los mismos. Cuando se efectúa una consulta interactúa con el ISRIModel parar obtener la lista de documentos con su respectivo índice de similitud, los ordena y los muestra utilizando el título y una breve sección 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 consulta 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 Boolean-ModelUtils.cs del subproyecto SRI.

En el modelo se permite también el trabajo con consultas en lenguaje natural (no lógicas) pero estas serán tratadas como una concatenacón de las palabras presentes en la consulta con el operador lógico " and ".

6. Modelo Vectorial Generalizado

Una característica que destaca 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 van computando.

Realizar la operación de guardado de forma eficiente resultó también un proceso complejo, se utilizó el Span [3] de C#, que permite utilizar un espacio consecutivo de memoria y convertir de un struct básico a otro a conveniencia en la interpretación de los datos almacenados. Al hacer provecho de estas facilidades para escribir en un archivo todo el contenido de este bloque de memoria, se logra crear el Storage del corpus en un tiempo promedio de 1.20 minutos para 18 000 documentos y obtener resultado para consultas en 8.0 segundos.

7. Evaluación de los modelos

Para evaluar los modelos, como las colecciones de NewsGroup y Reuters no tienen set de consultas y relevancias, se utiliza entonces la colección de Cranfield, además se usa Vaswani que es otra colección presente en la librería ir-dataset[4] 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 resultados según precisión, recobrado y medida F presentes en la librería ir-measures[5] que permite tener estos valores para distinta cantidad de documentos devuletos y tomando en cuenta distintos grados de relevancia.

7.1. Resultados de las Evaluaciones

Para la evaluación de Cranfield se tuvieron en cuenta diferentes grados de relevancia, como los resultados esperados de las consultas en el archivo qrel son valores entre -1 y 4 en orden ascendente de interés; se calculó la precisión, el recobrado y la medida F1 tomando niveles de relevancia 1, 2 y 3. Además la librería ir-measures tiene para cada medida de evaluación un parámtero judged_only booleano, que si está en true no considera los documentos que no fueron evaluados en el qrel y si es false considera a estos docuentos como no relevantes.

Rel	J-Only	Precisión	Recobrado	Medida F1
Vectorial				
1	T	0.2372	0.9986	0.3833
2	T	0.2182	0.9542	0.3552
3	T	0.1610	0.9047	0.2734
1	F	0.1240	0.5716	0.2037
2	F	0.1112	0.5361	0.1842
3	F	0.0774	0.4625	0.1327
Booleano				
1	T	0.2371	0.9986	0.3833
2	T	0.2182	0.9542	0.3552
3	T	0.1614	0.9055	0.2740
1	F	0.0274	0.1319	0.0453
2	F	0.0238	0.1287	0.0402
3	F	0.0182	0.1261	0.0318
Vectorial Generalizado				
1	T	0.2371	0.9986	0.3833
2	T	0.2182	0.9542	0.3552
3	T	0.1617	0.9058	0.2745
1	F	0.0048	0.0198	0.0078
2	F	0.0047	0.0203	0.0076
3	F	0.0028	0.0161	0.0047

Figura 1. Comparación de modelos-relevancia para Cranfield

Para Vaswani no pudimos tener diferentes grados de relevancia pues todos los qrel son de valor 1. Por lo que se procede a analizar la 20-Precisión, 30-Precisión y 50-Precisión para ambos modelos.

Modelo	J-Only	20-Precisión	30-Precisión	50-Precisión
Cranfield				
V	T	0.3477	0.2371	0.1432
V	F	0.1571	0.1240	0.0862
B	T	0.3477	0.2371	0.1432
B	F	0.0317	0.0274	0.0219
VG	T	0.3477	0.2371	0.1432
VG	F	0.0048	0.0048	0.0048
Vaswani				
V	T	0.7166	0.6003	0.4197
V	F	0.2274	0.1924	0.1584
B	T	0.7166	0.6003	0.4197
B	F	0.0731	0.0691	0.0544
VG	T	0.7166	0.6003	0.4197
VG	F	0.0059	0.0057	0.0040

Figura 2. Comparación de modelos-NPrecisión para Cranfield y Vaswani

Además se hace un análisis para la precisión, el recobrado y la medida F1 para las dos colecciones.

Modelo	J-Only	Precisión	Recobrado	Medida F1
Cranfield				
V	T	0.2371	0.9986	0.3833
V	F	0.1240	0.5716	0.2037
B	T	0.2371	0.9986	0.3833
B	F	0.0274	0.1319	0.0453
VG	T	0.2371	0.9986	0.3833
VG	F	0.0048	0.0198	0.0078
Vaswani				
V	T	0.6003	0.9232	0.7276
V	F	0.1924	0.3088	0.2371
B	T	0.6003	0.9232	0.7276
B	F	0.0691	0.0879	0.0774
VG	T	0.6003	0.9232	0.7276
VG	F	0.0057	0.0095	0.0071

Figura 3. Comparación de modelos-medidas de evaluación para Cranfield y Vaswani

Con los resultados de las medidas, se evidencia que el mejor de los tres modelos es sin dudas el modelo Vectorial, tanto para la colección de Cranfield como para la colección de Vaswani, de igual forma el modelo que peores resultados de-

vuelve es el Vectorial Generalizado, el cual además tiene en su contra la demora en realizar consultas y la necesidad de almacenar externamente los vectores de los documentos en archivos.

8. Cómo ejecutar el programa

Para ejecutar el código se tienen que seguir los siguientes pasos:

1. Crear una carpeta **contents** en el directorio 3Models-SRI y otra carpeta **DocSave**.
2. En **contents** crear 3 subcarpetas **Cran**, **20news** y **Reuters**.
3. En cada subcarpeta poner los documentos correspondientes, en el caso de 20news poner todas las carpetas que contienen los documentos, en Reuters poner los 22 archivos .sgm y en Cran el cran.all.1400.
4. Entrar a la carpeta UI y en consola ejecutar el comando **dotnet run**.
5. Abrir el navegador en la dirección indicada por el puerto de escucha que aparece en consola.
6. Seleccionar las colecciones deseadas y empezar a probar consultas.

Se requiere NET6.0 o superior, 4Gb de RAM disponibles al momento de la ejecución y al menos 13Gb de almacenamiento para la información procesada de los datos.

9. 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 Span, 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. Se comparó los resultados obtenidos en los modelos atendiendo a medidas de calidad, resultando el modelo Vactorial el de mejores resultados.

Referencias

1. StreamReader. URL: https://learn.microsoft.com/en-us/dotnet/api/system.io.streamreader?view=net-7.0. Consultado en 16 de diciembre de 2022.
2. Porter Stemming. URL: https://tartarus.org/martin/PorterStemmer/. Consultado en 16 de diciembre de 2022.
3. Span¡T¿Estructura. URL: https://learn.microsoft.com/es-es/dotnet/api/system.span-1?view=net-7.0. Consultado en 16 de diciembre de 2022.

4. ir-measures Documentation. URL: https://ir-datasets.com/index.html. Consultado el 16 de diciembre de 2022.
5. ir-measures Documentation. URL: https://ir-measur.es/en/latest/. Consultado el 16 de diciembre de 2022.