Sistema de Recuperación de Información

Carlos Rafael Ortega Lezacano and Eric Martín García
Grupo C511

Universidad de la Habana

Abstract. The abstract should summarize the contents of the paper and should contain at least 70 and at most 150 words. It should be written using the *abstract* environment.

Keywords: We would like to encourage you to list your keywords within the abstract section

1 Introducción

2 Diseño del Sistema

Un sistema de recuperación de información (IRS) se compone por el cuadruplo $< D,\ Q,\ F,\ R(q_j, d_j) >$, donde D es un conjunto de representaciones de los documentos, Q es un conjunto compuesto por representaciones lógicas de los pedidos que el usuario realiza al sistema, F es un framework para modelar las representaciones y R es una función de orden donde a cada consulta q_j y un documento d_j le asigna un valor acorde a la relevancia del documento para esa consulta. El proceso de interacción de un sistema con un usuario sigue el siguiente comportamiento: Primeramente el usuario realiza una consulta (un elemento de Q), esta pasa al motor de búsqueda, este es la componente más importante del sistema ya que en este se realizan las representaciones de los documentos[1], además de representar la consulta en formato interno, el sistema dará como resultado una lista ordenada por relevancia de los documentos del corpus. Analizaremos a continuación el procesador de consultas, como se compone el motor de búsqueda y finalmente la salida del sistema. En nuestro caso el sistema va enfocado en la búsqueda de documentos, un documento tiene varias características, dos de las más comunes y útiles a la hora de realizar búsquedas es el título y el cuerpo del documento.

Motor de Búsqueda

El motor de búsqueda contiene el modelo de recuperación de información, además en este se realiza el preprocesado de los documentos y su representación interna,

[1] En nuestro caso es un IRS vectorial por tanto la representación interna se realiza mediante vectores cuyas componentes son pesos asociados a la entidad resultante del preprocesamiento

también cada consulta se almacena en su forma interna para ser empleada en otras tareas. Además contiene la función de ranking la cual clasifica la relevancia de los documentos del corpus acorde a la consulta y el IRS.

Preprocesamiento: Para representar los documentos primeramente es necesario realizar un preprocesado, en nuestro sistema procedemos de forma simple empleando recursos del procesamiento de lenguaje natural. Las siguientes transformaciones son realizadas para cada documento del corpus:

1. **Tokenizar el documento:** Primeramente es necesario crear la representación básica de un documento, la división en tokens, estos son unidades básicas, de esta forma podemos realizar procesamiento más avanzado sobre el texto, ya desde esta parte del procesamiento podemos llevar a minúscula todas las palabras, eliminar símbolos y signos de puntuación. También es posible convertir los números a texto, de esta forma no tendremos problemas a la hora de construir los diversos modelos.

2. **Eliminación de *stopwords*:** Las *stopwords* como sabemos son palabras vacias de significado, las cuales pueden servir de nexo entre entidades o funcionan como modificadores. Si incluimos estas palabras en la representación del documento se afecta la efectividad del modelo debido a la alta frecuencia que poseen estas palabras en los textos.

3. **Stemming:** Este método busca relacionar palabras con igual significado pero que difieren en cuanto a la escritura, por la aplicación de prefijos o sufijos, se encuentran en plural o singular entre otras diferencias, en este caso no usaremos lematización que es común que se emplee luego del stemming, con remover las partes correspondientes es suficiente para obtener buenos resultados.

Estas técnicas son las empleadas por el sistema para obtener un conjunto de textos formados por entidades con un significado y peso correcto para empezar a construir el modelo, pero antes de definir el modelo es necesario definir como serán estas entidades y que representación interna tendrán. Se eligió emplear los tokens como entidad del sistema, de esta forma se construye una colección con el siguiente formato:

Representación del titulo y el cuerpo del documento preprocesado

```
corpus: {
                [id: n,
                'title': Tokenize(doc_title),
                'body': Tokenize(doc_body),
                ],
                ...
        }
```

Vectorización: De esta forma contamos con el identificador para cada documento del corpus, su titulo y cuerpo preprocesado. Pero todavia tenemos tokens los cuales no pueden ser la entrada para el modelo, es necesario realizar una vectorización empleando TF-IDF. Para calcular los valores de frecuencia emplearemos el titulo y el cuerpo del documento juntos, sin realizar distinción, o sea el titulo podría ser una oración más del texto.

TF-IDF: Es una técnica para cuantificar una palabra en documentos, generalmente calculamos un peso para cada palabra que significa la importancia de la palabra en el documento y corpus. Este método es una técnica ampliamente utilizada en recuperación de información y minería de textos. Esta se representa por $w(t_i, d_j)$ donde t_i es un término del documento d_j, en nuestro caso un los t_j son tokens.

El TF-IDF se compone de dos conceptos fundamentales que se apoyan en el procesamiento de los documentos, el TF (Frecuencia de Termino) y el IDF (Frecuencia Inversa de Documento).

TF: Mide la frecuencia de una palabra en un documento. Esto depende en gran medida de la longitud del documento y de la generalidad de la palabra, por ejemplo, una palabra muy común como "casa" puede aparecer varias veces en un documento. Como los documentos pueden tener una longitud variable entonces contar simplemente las palabras podría dar prioridad a documentos de mayor tamaño, por eso es que se divide la cantidad de ocurrencias por el número total de palabra del documento $N(d_j)$. TF es individual para cada documento y palabra, por lo tanto, podemos formular TF de la siguiente manera:

$$f_s(t_i, d_j) = \frac{C(t_i, d_j)}{N(d_j)} \tag{1}$$

Donde $C(t_i, d_j)$ representará la cantidad de ocurrencias del termino en el documento. Recordemos que estamos vectorizando los documentos por tanto es necesario introducir el concepto de **vocabulario**, este es un conjunto formado por todas las palabras del corpus, o sea todas las palabras que hay en cada documento, de esta forma garantizamos que los vectores para cada documento tengan siempre las misma dimensión, donde cada componente tendrá el valor de frecuencia asociado para esa palabra, este estará en el intervalo $[0, 1]$. Aunque empleando esta frecuencia es posible obtener resultados, todavía podemos mejorar los valores para el vector. Antes de definir IDF es necesario definir Frecuencia de Documento, ya que su inversa es IDF.

DF: Esto mide la importancia del documento en todo el conjunto de corpus, esto es muy similar a TF. La única diferencia es que TF es el contador de frecuencia para un término t en el documento d, donde DF es el recuento de apariciones del término t_i en el conjunto de documentos D. En otras palabras, DF es el número de documentos en los que está presente la palabra. Consideramos una ocurrencia

si el término consta en el documento al menos una vez, no necesitamos saber el número de veces que el término está presente.

Como en el caso de TF también normalizamos dividiendo por el número total de documentos. Nuestro principal objetivo es conocer la informatividad de un término, para eso invertimos DF.

IDF: Es la inversa de la frecuencia del documento que mide la informatividad del término t. Cuando calculemos el IDF, será muy bajo para las palabras más frecuentes. Como es posible que el DF sea cero entonces es necesario realizar algunas modificaciones para no dividir entre 0, además si el corpus es grande podemos entonces obtener un valor que no sea de utilidad por lo tanto emplearemos para calcular el IDF.

$$\overline{d}_s(t_i) = \log \frac{|D|}{d_s(t_i) + 1} \tag{2}$$

Ahora combinemos ambas frecuencias, de esta forma tendremos una relación entre términos de mucha ocurrencia con respecto a aquellos que de baja frecuencia en los documentos.

$$w(t_i, d_j) = f_s(t_i, d_j) \log \frac{|D|}{d_s(t_i) + 1} \tag{3}$$

Hasta el momento se ha expuesto el concepto de TF-IDF para documentos pero ignorando si tienen titulo o no. Los textos que conforman el corpus del motor de búsqueda tienen diversas características y hemos elegido el titulo y el cuerpo para representar, por lo tanto debemos determinar cual de estos es más importante, o sea para dar un valor de relevancia a un documento que tiene mayor peso el título o el cuerpo, para ello definamos el parámetro α el cual define cuanto peso le damos al título a la hora de calcular la relevancia, de esta forma tendríamos que determinar los valores de TF-IDF tanto para el título como para el cuerpo, relacionando estos valores con α para obtener la frecuencia final para el término en el documento. Como los valores de TF-IDF para un término no toman en cuenta si este se encontraba en el titulo del documento o en el cuerpo, como se expreso anteriormente entonces definamos w_t como el valor de TF-IDF para un término en el título y w_b para cuando se encuentra en el cuerpo del documento. Finalmente obtenemos la expresión.

$$w(t_i, d_j) = w_t(t_i, d_j)\alpha + w_b(t_i, d_j)(1 - \alpha), \qquad \alpha \in [0, 1] \tag{4}$$

Podemos notar como el valor de α es un coeficiente asociado a las frecuencias de cada parte del documento, además un aspecto importante a tener en cuenta es lo siguiente: Si el termino t_i aparece tanto en el título como en el cuerpo del documento entonces $w_t(t_i, d_j) = w_b(t_i, d_j)$, debido a que no se tomó en cuenta la posición a la hora de calcular los valores de frecuencia, por lo tanto el valor final sería:

$$w(t_i, d_j) = w_b(t_i, d_j) \tag{5}$$

Ya tenemos el valor de frecuencia además de que empleamos el parámetro α para definir la importancia que tiene el título o no al momento de determinar la relevancia, pasemos entonces a vectorizar los documentos. Para vectorizar el documento emplearemos un Bag of Words.

Bag of Words: Este consiste en tomar todas las palabras del corpus que constituyen el vocabulario (V) y formar una colección donde cada palabra tiene asociado un índice, de esta forma obtenemos un vector $\overrightarrow{d_j}$ de dimension $1 \times |V|$ asociado al documento d_j, donde en cada componente se encuentra el valor de TF-IDF asociado a la palabra.

De esta forma obtenemos para cada documento del corpus el vector $\overrightarrow{d_j}$, ahora pasemos a determinar como se procesa la consulta para obtener su vector correspondiente y dependiendo de este que función de ranking se empleó.

Consultas

Como sabemos en el sistema existe un conjunto Q que contiene las consultas realizadas por el usuario, una consulta q no es más que una oración u entidad empleada para realizar la búsqueda. Para poder determinar cuales documentos son relevantes para esa consulta es necesario establecer un orden de los mismo acorde a la importancia por lo tanto debemos convertir la consulta a un vector empleando el mismo proceso que vimos en el motor de búsqueda.

Se usa el mismo vocabulario V empleado para la representación del corpus, y se procese a crear el vector $\overrightarrow{q_i}$, en caso que existan palabras en q_i que no se encuentran en el vocabulario serán ignoradas.

El valor que tomarán las componentes no tiene necesariamente que ser calculo igual a como se realiza en el motor de búsqueda para representar el corpus, puede depender de la función de ranking o si usamos un modelo un poco más complejo.

Función de Ranking

Salida del Sistema

3 Sobre la implementación

4 Evaluación del Sistema

5 Resultados Obtenidos

6 Recomendaciones

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)

2. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
6. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`