


Sistemas de Recuperación de Información

Motor de
búsqueda:
Sherlock 

Autores

Laura Victoria Riera Pérez

Leandro Rodríguez Llosa

Marcos Manuel Tirador del Riego

Resumen Se abordan los aspectos principales de una posible implementación del modelo vectorial clásico.

Palabras clave — recuperación de información (RI) · modelo vectorial

Abstract. Se abordan los aspectos principales de una posible implementación del modelo vectorial clásico.

Keywords — recuperación de información (RI) · modelo vectorial

Índice general

1	Introducción	1
2	Diseño del sistema	1
2.1	Documentos	1
2.2	Normalización de un término	1
2.3	Corpus	1
2.4	Modelo base	2
3	Modelo Booleano	2
3.1	Descripción	2
3.2	Implementación	2
4	Modelo Vectorial	2
4.1	Preprocesamiento	2
4.2	Recuperación de documentos	2
5	Modelo Fuzzy	2
5.1	Descripción del modelo usado	3
5.2	Implementación.	4
6	Evaluación de los modelos	5
6.1	Modelo Fuzzy	5
7	Retroalimentación	5
8	Agrupamiento	5
8.1	K-means	6
8.2	Objetivo perseguido	6
8.3	Implementación	7
8.4	Resultados	7
9	Conclusiones	8

1 Introducción

En la actualidad, con el inmenso crecimiento del internet, se convierte en un reto cada vez mayor el manejo de la información, su recuperación y la extracción de conocimiento de ella. Por esto, es de especial interés la creación de algoritmos que ayuden a su manipulación. En este informe se expondrán algunas ideas importantes seguidas a la hora de implementar un modelo de recuperación de información clásico: el modelo vectorial.

2 Diseño del sistema

2.1 Documentos

Para el desarrollo de este Sistema de Recuperación de la Información modelamos un documento como un objeto que contiene al menos dos propiedades, un `doc_id` que identifica de manera única a un documento dentro del conjunto de documentos del dataset en cuestión; y un `text` que corresponde con el texto de este. También puede tener otras propiedades, por ejemplo: `title`, `author`; pero eso depende de la riqueza del dataset que provee el paquete de Python `ir_datasets` (ver [3]).

Como el texto de un documento es incómodo de manipular por venir en forma de `string`, este se tokeniza y convierte en una lista de términos indexados normalizados. Esto se logra haciendo uso del paquete de Python `re` (referirse a [4]) que proporciona una colección de funciones que facilitan el trabajo con expresiones regulares.

2.2 Normalización de un término

Es importante destacar algunas asunciones que se tuvieron en cuenta en el proceso de tokenización. No se considera relevante la diferenciación entre una letra mayúscula y una minúscula, ya que, en la mayoría de los casos, el significado semántico que expresan es el mismo. Para reducir algunos errores ortográficos, y que esto no perjudique la recuperación de un documento importante, se considera que las tildes no diferencian una palabra de otra. Se conoce que esto último no es cierto en el español, pero por el momento se está trabajando con textos en inglés.

Resumiendo, se trata de llevar todos los términos a cadenas de caracteres que contienen letras minúsculas o números.

2.3 Corpus

Un corpus no es más que el conjunto de documentos de un dataset, tokenizados y normalizados.

2.4 Modelo base

Se define un modelo base como un concepto abstracto, que engloba los comportamientos comunes que debe tener cada modelo de recuperación de información.

Cada instancia de un modelo tiene un corpus asociado a este, sobre el cual se hacen las búsquedas. Se deja en manos del programador qué preprocesamientos hacer con el corpus, en dependencia del modelo que se esté implementando.

3 Modelo Booleano

3.1 Descripción

3.2 Implementación

4 Modelo Vectorial

4.1 Preprocesamiento

Según la fórmula de similitud del coseno ([1, Ecuación (6.10)]), se puede notar que el aporte de un documento a la fórmula se mantiene invariante para todas las consultas, ya que este solo depende del vector que representa al documento, el cual es independiente de la consulta. Por tanto, como el corpus sobre el cuál se va a recuperar información se mantendrá estático, se calcula el peso de cada término en cada documento para así poder utilizarlo en cada consulta que se realice.

Para calcular el peso de cada término en cada documento según [1, Ecuación (2.3)], primero se necesita calcular las frecuencias normalizadas de los términos en cada documento (TF [2, Ecuación (2.1)]), y la frecuencia de ocurrencia de cada término dentro de todos los documentos del corpus (IDF [2, Ecuación(2.2)]).

4.2 Recuperación de documentos

La primera fase es similar a la del preprocesamiento de los documentos del corpus. Lo primero que se hace es tokenizar y normalizar la consulta. Luego se hallan los TFs, y se calcula el peso de cada término en la consulta. Para el cálculo de los pesos de cada término se utiliza la medida de suavizado $\alpha = 0.4$ para amortizar la contribución de la frecuencia del término (ver [2, ecuación (2.4)]). Por último se halla la cercanía entre el vector consulta y cada vector documento, utilizando la similitud del coseno entre los vectores según [1, Ecuación 6.10], y se hace un ranking teniendo este valor calculado.

5 Modelo Fuzzy

En el modelo booleano se representan las consultas y documentos como conjuntos de palabras. Al determinar que un documento es relevante a una consulta si y solo si tiene una coincidencia exacta de las palabras en la consulta solamente

nos acercamos parcialmente al contenido semántico real de los contenidos. Una alternativa sería definir un conjunto difuso por cada palabra de la consulta y determinar un grado de pertenencia de cada documento a este conjunto, para luego basado en esto, poder asignar un grado de relevancia de cada documento a la consulta dada. Esta es la idea básica detrás de distintos modelos de recuperación de la información basados en conjuntos difusos.

Se presentan a continuación algunos conceptos de la teoría de conjuntos difusos necesario para entender el modelo implementado que se describe en esta sección.

Definición 1 ([2, Section 2.6.1]) *Un conjunto difuso A de un universo de discurso U está caracterizado por una función de membresía $\mu_A : U \rightarrow [0, 1]$ que asocia a cada elemento $u \in U$ un número $\mu_A(u)$ en el intervalo $[0, 1]$.*

Las tres operaciones más usadas de conjuntos difusos se definen a continuación.

Definición 2 ([2, Section 2.6.1]) *Sea U el universo de discurso, A y B dos conjuntos difusos de U y \bar{A} el complemento de A en U . Sea además un elemento $u \in U$. Entonces,*

$$\begin{aligned}\mu_{\bar{A}}(u) &= 1 - \mu_A(u) \\ \mu_{A \cup B}(u) &= \max(\mu_A(u), \mu_B(u)) \\ \mu_{A \cap B}(u) &= \min(\mu_A(u), \mu_B(u)).\end{aligned}$$

5.1 Descripción del modelo usado

El modelo fuzzy que se seleccionó para su implementación fue propuesto por Ogawa, Morita y Kobayashi en [5]. En este los autores se basan en el uso de la relación entre términos para expandir los términos de las consultas, de forma que se encuentren más documentos que sean relevantes a las necesidades del usuario. Para expandir las consultas se opta por el uso de la matriz de conexión de términos clave (keyword connection matrix). En esta matriz se encuentra la correlación normalizada entre cualesquiera dos términos k_i y k_j definida como

$$c_{i,j} = \frac{n_{i,j}}{n_i + n_j - n_{i,j}},$$

donde n_i (respectivamente n_j) es el número de documentos que contienen a k_i (respectivamente k_j) y $n_{i,j}$ es el número de documentos que los contienen a ambos. Esta correlación se basa en la idea de que si dos palabras están relacionadas aparecerán, con frecuencia, juntas en un mismo documento.

Definimos entonces la pertenencia de un documento d_j al conjunto difuso relativo al término k_i como

$$\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l}).$$

Como podemos ver, luego de expandir el producto de la derecha, esta fórmula lo que computa es cierta suma algebraica de la correlación entre k_l y todos los

términos del documento d_j . Se observa que esta expresión no es exactamente la suma directa de los $c_{i,l}$ para cada k_l , sino que es una suma suavizada. La expresión resultante de expandir es similar a un principio de inclusión exclusión. Esto da la idea de que la relación entre d_j y k_i no es la suma de las correlaciones entre cada $k_l \in d_j$ y k_i , sino que tiene en cuenta la relación conjunta con k_i de los subconjuntos de términos de d_j , ya que puede que varios k_l diferentes estén muy relacionados entre sí, y a k_i , y su aporte a la suma este siendo sobre valorado.

Aun con lo expuesto antes, cuando un término $k_l \in d_j$ está muy relacionado con k_i (es decir $c_{i,l} \approx 1$), entonces $\mu_{i,j} \approx 1$, lo cual dice que d_j tiene un grado de pertenencia alto al conjunto difuso de k_i . Lo opuesto sucede cuando todos los términos $k_l \in d_j$ están vagamente relacionados con k_i , en cuyo caso $c_{i,l} \approx 0$. Por tanto se mantiene el objetivo de la relación, aun cuando la suma usada no es la usual.

Las consultas del usuario en este modelo vienen dadas de la misma forma que en el modelo booleano. Estamos hablando de una expresión lógica que se convierte luego a forma normal disyuntiva. Supongamos entonces que una consulta q se descompone en las n componentes conjuntivas cc_1, cc_2, \dots, cc_n . Sea $\mu_{cc_t,j}$ el grado de pertenencia del documento d_j al conjunto de documentos relevantes para la forma normal disyuntiva c_t . Este se calcula como sigue

$$\mu_{cc_t,j} = \prod_{k_i \in cc_t} \mu'_{i,j},$$

donde $\mu'_{i,j} = 1 - \mu_{i,j}$ si k_i aparece negado en cc_t y $\mu'_{i,j} = \mu_{i,j}$ en otro caso. Entonces si denotamos $\mu_{q,j}$ como el grado de pertenencia del documento d_j al conjunto de documentos relevantes a la consulta q , este se podría calcular como

$$\mu_{q,j} = 1 - \prod_{t=1}^n (1 - \mu_{cc_t,j}).$$

Según la Definición 2, el valor $\mu_{cc_t,j}$ (el grado de pertenencia a una conjunto difuso conjuntivo) y $\mu_{q,j}$ (el grado de pertenencia a un conjunto difuso disyuntivo) deberían ser calculados tomando el mínimo y el máximo de las variables que intervienen respectivamente. Sin embargo se opta en este caso por usar producto y suma algebraica, respectivamente, para suavizar los resultados.

5.2 Implementación.

Para implementar el modelo descrito los autores del presente se basaron en el procesamiento de los documentos y consultas que se expusieron en el modelo booleano, ya que estas adoptan la misma forma. Se modifica entonces el método que computa el resultado de una consulta, asignando en esta ocasión una puntuación a cada documento d_j respecto a la consulta q , correspondiente a $\mu_{q,j}$. Los documentos serán recuperados estableciendo un orden según este valor.

Para calcular los valores en cada consulta de $\mu_{q,j}$ simplemente computamos las fórmulas descritas antes. La correlación entre términos, sin embargo, se calcula una sola vez para el total de pares de términos que aparecen en todos los

documentos, y se guarda en el almacenamiento físico de la computadora. Esto se hace para no tener que recalculan todos estos valores más de una vez, ya que constituye un costo alto en tiempo.

6 Evaluación de los modelos

6.1 Modelo Booleano

6.2 Modelo Vectorial

6.3 Modelo Fuzzy

Al ser un modelo que extiende el modelo booleano, aunque resuelve muchas de las dificultades del anterior, este aun acarrea las dificultades del mismo. Una de estas principales dificultades es que el lenguaje de consultas es complejo para inexpertos. Las conjuntos de consultas de prueba tanto de *Cranfield* como de *Vaswani* no están pensadas para aplicarlas sobre un modelo booleano. Esto se puede apreciar del simple hecho de que las mismas constituyen oraciones y no expresiones lógicas.

Sin embargo, si se puede apreciar una ligera mejoría respecto al modelo booleano (en el caso de *Cranfield* se obtiene un valor de la F-medida casi cuatro veces mayor). Este resultado es de esperar ya que este modelo mejora algunas de las dificultades del anterior, como por ejemplo que en este caso la coincidencia de los documentos no tiene que ser exacta, se crea un ranking, se adiciona cierta semántica a los términos analizando cierta correlación entre los pares de ellos y además, de esto último se puede inferir que no todos los términos seguirán siendo igual de importantes.

En el caso del conjunto de datos *cord19* este no pudo usarse en el modelo fuzzy ya que la gran cantidad de documentos y términos que posee este, es muy grande, lo que hace que calcular la correlación entre cada par de términos incurra en un uso de recursos muy elevado. Si dicha correlación se precalcula, harían falta más de 20GB de RAM según los cálculos hechos (que no se presentan pues son aproximaciones poco formales). Si en cambio se hace en el momento de cada consulta solo para los términos de la consulta, ejecutar cada consulta tomaría más de 5 minutos. Además, el modelo fuzzy no ha sido extensamente probado en experimentos con colecciones grandes de documentos¹.

7 Retroalimentación

8 Agrupamiento

Los algoritmos de agrupamiento, como su nombre lo indica, agrupan un conjunto de documentos en subconjuntos o clústeres. Los grupos formados deben tener un alto grado de asociación entre los documentos de un mismo grupo, es

¹ Se tomó de [2], página 38.

decir, deben ser lo más similares posibles, y un bajo grado entre miembros de diferentes grupos. Es la forma más común de *aprendizaje no supervisado*, es decir no hay ningún experto humano que haya asignado documentos a clases. Es la distribución y composición de los datos lo que determinará la pertenencia al clúster.

Se presenta a continuación la hipótesis en que se basan los algoritmos de agrupamiento, que fue tomada de [1, Sección 16.1].

Hipótesis de agrupamiento:² *Los documentos en el mismo grupo se comportan de manera similar con respecto a la relevancia para las necesidades de información.*

La hipótesis establece que si hay un documento de un grupo que es relevante a una solicitud de búsqueda, entonces es probable que otros documentos del mismo clúster también sean relevantes.

8.1 K-means

En este trabajo se optó por usar uno de los algoritmos de agrupamiento más importantes, conocido como K-means. Este algoritmo se ejecuta sobre un conjunto espacio de documentos representados como vectores dimensionales. El mismo fija inicialmente de forma aleatoria los centroides de los clústeres y en cada iteración los va moviendo de forma que se disminuya en cada paso la suma de los cuadrados de las distancias de cada documento a su clúster más cercano (RSS).

Se define el centroide de un clúster formalmente como

$$\vec{\mu}(w_k) = \frac{1}{|w|} \sum_{x \in w_k} \vec{x},$$

donde w_k es el clúster y \vec{x} es un vector que representa a un documento que pertenece a w . Se define RSS como

$$RSS = \sum_{i=1}^K \sum_{\vec{x} \in w_k} |\vec{x} - \vec{\mu}(w_k)|^2,$$

donde K es el número de clústeres.

8.2 Objetivo perseguido

Se decidió usar este algoritmo de agrupamiento para mejorar la recuperación de documentos en el modelo vectorial. La idea perseguida es que según la hipótesis de agrupamiento se espera que documentos en el clúster más cercano al vector de la consulta debe contener con mayor probabilidad documentos similares a

² Las ideas de esta introducción a la sección fueron tomada de [6] de los propios autores de este informe.

la consulta. Entonces reordenamos y reasignamos reasignamos puntuaciones de relevancia a los documentos, de forma que sean más relevantes aquellos que pertenezcan a clústeres más cercanos a la consulta, manteniendo el orden relativo que tenían antes aquellos documentos en el mismo clúster. Además, podemos presentar al usuario los documentos agrupados en los clústeres, siendo más fácil escanear algunos grupos coherentes que muchos documentos individuales. Esto es particularmente útil si un término de búsqueda tiene diferentes significados.

8.3 Implementación

Para poner en práctica la idea perseguida, se implementó un nuevo modelo de recuperación de la información que se sustenta en el modelo vectorial antes explicado. Este modifica el método de recuperación de documentos a una consulta para añadir el criterio por clústeres descrito. Los autores se auxiliaron de la biblioteca de python *sk-learn*.

Una de las decisiones de implementación que se tuvo que tomar fue la elección de un número k de clústeres adecuados. Se implementó primero el conocido método del codo que toma la decisión en función de la gráfica de los valores de RSS para cada elección de k considerada. Sin embargo este método no arrojó ninguna luz en el conjunto de datos de *Cranfield* el cual se usó para implementar este modelo. Entonces se optó por una decisión en función de minimizar los RSS, pero penalizando el número de clústeres usados. La expresión que se utilizó fue la siguiente

$$k = \arg \min_{k \in \mathbb{N}} (RSS_k + \lambda * k),$$

donde λ se tomo como 0,08 multiplicado por la cantidad de documentos del corpus. Según esta expresión un buen número de clústeres (inferior a 10 porque para valores mucho más grandes el algoritmo tomaba mucho tiempo en ejecutarse) es $k = 5$, y este es el utilizado.

Para determinar la nueva puntuación de cada documento respecto a una consulta se toma

$$score_d = \frac{oldscore_d}{10^6} + \frac{1}{|\vec{q} - \vec{\mu}_d|},$$

donde \vec{q} es el vector consulta y $\vec{\mu}$ es el centroide al que pertenece el documento d . De esta forma siempre obtienen mejor puntuación los documentos de los clústeres más cercanos a la consulta y se mantiene el orden relativo que tenían los documentos de un mismo clúster según el modelo vectorial.

8.4 Resultados

En su aplicación en el conjunto de datos de *Cranfield*, y su evaluación, este modelo tuvo resultados muy similares a los del modelo vectorial. Por tanto, una conclusión a la que arribamos es que la aplicación de K-means en la forma descrita no constituye una mejora del modelo vectorial. Estos resultados son coherentes con la idea intuitiva de que si los documentos recuperados con el

nuevo modelo son todos de un mismo clúster, entonces son cercanos entre sí, y por tanto cercanos al vector consulta, por lo que deberían ser recuperados también en el modelo vectorial clásico.

Sin embargo, se tienen otras ideas de como aplicar este algoritmo al mismo modelo, sobre la base del mismo código, que se seguirán probando en busca de mejorar los resultados.

9 Conclusiones

Este trabajo presenta una propuesta para la modelación del problema de recuperación de información. Se detallaron aspectos generales del modelo vectorial clásico, así como decisiones de diseño específicas de la propia interpretación del problema.

Referencias

1. Maning C. D.: *An Introduction To Information Retrieval* (2009).
2. Ricardo Baeza-Yates: *Modern Information Retrieval* (1999).
3. Documentación oficial: <https://ir-datasets.com/index.html>
4. Documentación oficial: <https://docs.python.org/3/library/re.html>
5. Y. Ogawa, T. Morita y K. Kobayashi. A fuzzy document retrieval system using the keyword connection matrix and a learning method. *Fuzzy Sets and Systems*, 39:163-179, 1991.
6. Laura Riera Pérez y Marcos Tirador del Riego. Aplicaciones del agrupamiento y de la clasificación en la recuperación de información en la Web.