UNIVERSIDAD
COMPLUTENSE
MADRID

Modelos de recuperación de información

Sistemas de Gestión de Datos y de la Información
Enrique Martín - emartinm@ucm.es
Máster en Ingeniería Informática
Fac. Informática

Modelos de recuperación de información

- Los modelos de recuperación de información (MRI) formalizan el proceso de **encajar** una **consulta** con un **documento**.
- Definición: un modelo de recuperación está compuesto por:
  - **D**: vista lógica de los documentos en la colección
  - **Q**: vista lógica de las consultas de los usuarios
  - **R(d,q)**: función que relaciona un número real a un documento *d* y una consulta *q*.

Modelos de recuperación de información

- Los documentos reales son secuencias de caracteres o bytes, pero para el MRI pueden adoptar otras formas: tuplas binarias, vectores n-dimensionales con valores reales.
- Igualmente, las consultas son secuencias de términos clave con conectores, pero para el MRI se representan de otra formas como tuplas binarias o vectores.
- La función R tomará como entrada documentos y consultas en su visión lógica.

Modelos de recuperación de información

- Hay varios tipos de MRI:
  - **Modelo booleano**
  - **Modelo vectorial**
  - Modelo probabilista
  - Modelo booleano extendido
  - Modelo *fuzzy*

Conceptos básicos

- Un **término clave** es una palabra o una secuencia de palabras de un documento. Los identificaremos con *k*.
- Sea *t* el número de términos clave distintos en una colección. Entonces el **vocabulario** estará formado por V = {*k1*, *k2*, ..., *kt*}
- Si el número de documentos es *n* entonces se identifican como *d1, d2, ..., dn*.

Modelo booleano

Modelo booleano

- Las consultas se representarán como fórmulas booleanas:
  - Consultas (**c**) ::= **k** | **c** AND **c** | **c** OR **c** | NOT **c**
  - Los símbolos terminales son los posibles términos clave **k**.
  - Las conectoras serán las usuales: AND, OR, NOT
- Ejemplos:
  - deporte OR ocio
  - (deporte OR ocio) AND agua
  - pesca AND mar AND (NOT rio)

Modelo booleano

- El modelo booleano devolverá aquellos documentos de la colección que **encajen completamente** con la consulta, es decir, que contengan (o no contengan) los términos clave solicitados.
- Esto hace que documentos que satisfacen **parcialmente** la consulta (aunque sea en gran medida) sean **descartados**.

Patrones de co-ocurrencia

- Si en un documento *di* aparecen únicamente los términos *kl*, *km* y *kn* diremos que se observa el **patrón de co-ocurrencia** [kl, km, kn] en el documento *di*.
- Si el vocabulario tiene longitud t, existirán 2^t patrones de co-ocurrencia. Al conjunto de todos los patrones de co-ocurrencia posibles lo denotaremos **K**.
- Para referirnos a un patrón de co-ocurrencia utilizaremos la letra **p**.

Patrones de co-ocurencia

- Los patrones de co-ocurrencia se representan mediante tuplas:
  - (1,0,0,...,0) representa que únicamente se observa el primer término clave
  - (1,1,0,...,0) representa que únicamente se observa el primer y el segundo término clave
  - (1,1,1,...,1) representa que se observan todos los términos clave.
- Cada una de estas tuplas también se llama **componente conjuntiva de términos clave**.

Patrones de co-ocurencia

- Para cada documento *di* asociamos un patrón de co-ocurrencia **c(di)** que describe los términos clave que aparecen en el documento.
- De la misma manera, a cada consulta *q* se le asocian **uno o varios patrones de co-ocurrencia** indicando los términos clave que contiene. En general **c(q)** devuelve un subconjunto de **K.**

Patrones de co-ocurencia

- ¿Por qué asignamos varios patrones de co-ocurrencia a una consulta? Porque pueden existir varios patrones de co-ocurrencia que lo cumplan.
- ***Ejemplo***. con t = 3 y q = (k1 AND k2) OR k3 tenemos que hay varios patrones de co-ocurrencia:
  - (1,1,0) y (1,1,1) para hacer cierto (k1 AND k2)
  - (0,0,1), (0,1,1), (1,0,1) y (1,1,1) para k3

Similitud booleana

- Para el modelo booleano, la función de similitud **R(d,q)** que relaciona documentos y consultas devuelve un valor en {0,1}, y está definida de la forma:

$$R(d,q) = \begin{cases} 1 & si\ c(d) \in c(q) \\ 0 & e.o.c \end{cases}$$

- Si **R(d,q)** = 1, entonces el documento es relevante, si **R(d,q)** = 0 entonces no es relevante.

Similitud booleana: ejemplo

- Consideremos que **t**=3 y tenemos 4 documentos *d1*, *d2*, *d3* y *d4* tales que:
  c(d1) = (1,0,1) **c(d2)** = (0,1,1)
  c(d3) = (1,0,0) **c(d4)** = (0,1,0)
- Tenemos la consulta **q** = (k1 AND k2) OR k3, donde **c(q)** = {(1,1,0), (1,1,1), (0,0,1), (0,1,1), (1,0,1)}
  - R(d1,q) = 1, porque (1,0,1) en c(q)
  - R(d2,q) = 1, porque (0,1,1) en c(q)
  - R(d3,q) = 0, porque (1,0,0) no en c(q)
  - R(d4,q) = 0, porque (0,1,0) no en c(q)

Similitud booleana

- La función **R(d,q)** simplemente nos devuelve un valor {0,1}, donde:
  - 1 → el documento **es relevante**
  - 0 → el documento **no es relevante**
- *Idea:* la función **R(d,q)** es un “todo o nada”
- Con esta función puedo partir el conjunto de todos los documentos en 2, y mostrar al usuario únicamente los relevantes.

Similitud booleana

- Sin embargo, según la función todos los documentos tales que **R(d,q)** = 1 son “igual de relevantes”. ¿En qué orden se los muestro al usuario?
- Puedo usar algún orden: fecha de creación/modificación, tamaño, popularidad... pero no estaría utilizando la relevancia para ordenar.
- **Resumen**: el modelo booleano es sencillo y eficiente pero algo limitado.

Modelo vectorial

Modelo vectorial

- Mitiga las limitaciones del modelo booleano.
- En el modelo vectorial tanto los documentos como las consultas se representarán como **vectores t-dimensionales de números reales**.
- La función de similitud calculará la similitud de documentos y consultas comparando los vectores, en concreto el **ángulo** que forman.

Modelo vectorial

- Al comparar los documentos y consultas mediante su ángulo, obtengo **valores continuos** (reales) como medida de la **similitud** en lugar de obtener los valores discretos {0,1}.
- Gracias a esto podremos **ordenarlos** por su valor de **relevancia**, y adicionalmente mostrar aquellos documentos con relevancia mayor que un determinado umbral (p.ej. > 0,7).

Modelo vectorial

- Los **vectores** que manejaremos serán de la forma:
  - $dj = (w_{1j}, w_{2j}, \ldots, w_{tj})$, donde w_{ij} será el peso del término clave i-ésimo en el documento j-ésimo.
  - $q = (w_{1q}, w_{2q}, \ldots, w_{tq})$, donde w_{iq} será el peso del término clave i-ésimo en la consulta q-ésima.
- Intuitivamente, el **peso** representa la **importancia del término clave** en el documento o la consulta.

Pesos

- Si un término clave aparece muchas veces en un documento, lo normal es que “precise” mejor el contenido del documento y por tanto sea importante.
- Si en un documento un término clave *ki* aparece 1 vez y otro *kj* aparece 50, *kj* es **más importante** que *ki*.
- El **peso** debería **cuantificar** esta **importancia por apariciones**.

Pesos

- Imaginemos una colección de 1000 documentos, donde en todos ellos aparece el término clave *k*. Este término no será importante porque no nos sirve para **discriminar** unos documentos de otros.
- Sin embargo, si *k*' aparece en solo 5 de ellos será muy importante porque reducirá bastante el conjunto de documentos relevantes.
- El **peso** debería **cuantificar** también esta capacidad **discriminante**.

Pesos

- Los pesos w_{ij} son valores reales positivos, es decir, $w_{ij} \geq 0$
- Como el peso de los distintos términos clave dependerá de las apariciones que tengan en los documentos, un concepto importante será la **frecuencia**.

Pesos: terminología

- La frecuencia f_{ij} es el número de apariciones del término clave *ki* en el documento *dj.*
- De manera similar, f_{iq} es la frecuencia del término clave *ki* en la consulta *q.*
- El número total de documentos es *N.*
- *ni* es el número de documentos en los que aparece el término clave *ki*. Por tanto siempre se tiene que $ni \leq N$.

Pesos TF-IDF

- Uno de los esquemas más populares para asignar pesos a los términos clave de cada documento es el conocido como TF-IDF (*term frecuency-inverse document frecuency*).
- Combina dos valores relativos a las frecuencias de los términos clave:
  - *Term frecuency*: frecuencia del término clave en el documento.
  - *Inverted document frecuency*: frecuencia inversa del término en todos los documentos.

Term frecuency

- La frecuencia de término clave *(term frecuency)* de un término clave *ki* en un documento *dj*, representada como tf_{ij} , se define como:

  $tf_{ij} = 1 + \log f_{ij}$, si $f_{ij} > 0$

  $tf_{ij} = 0$, si $f_{ij} = 0$ *(log 0 no está definido)*

- La frecuencia de término clave tf_{ij} será **proporcional** a la frecuencia de *ki* en el documento *dj*.

(Inciso)

No lo incluyo de manera explícita en la notación, pero en este tema todos los logaritmos son en **base 2:**

$$\log N = \log_2 N$$

Term frecuency: Ejemplo

- Imaginemos que **t**=3 y para un documento *dj* tenemos las frecuencias:

  $f_{1j} = 4$ $f_{2j} = 0$ $f_{3j} = 2$

- En este caso tendríamos los siguientes valores de frecuencia *de término clave* para el documento *dj*:
  - $tf_{1j} = 1 + \log 4 = 1 + 2 = \mathbf{3}$
  - $tf_{2j} = \mathbf{0}$ (porque $f_{2j} = 0$)
  - $tf_{3j} = 1 + \log 2 = 1 + 1 = \mathbf{2}$

Term frecuency

- Si queremos que sea proporcional, ¿por qué no utilizar directamente $tf_{ij} = f_{ij}$? ¿Por qué añadimos el logaritmo?
  - 1) Para amortiguar, ya que un término que aparece el doble no tiene por qué ser el doble de importante.
  - 2) Para que la magnitud sea comparable de manera directa con la frecuencia inversa de documento (IDF).

Inverse document frecuency

- La **frecuencia inversa de documento**, idf_i, trata de reflejar la importancia del término clave *ki* dentro de la colección.
- Si un término clave aparece en **muchos documentos** de la colección, es **poco discriminante** y queremos que su **peso** sea **bajo**.
- Si aparece en **pocos documentos**, es **bastante discriminante** y queremos que tenga un **peso alto**.

Inverse document frecuency

- La **frecuencia inversa de documento** del término clave *ki* está definida mediante la siguiente fórmula:

  $$\mathbf{idf_i = log\ (N/ni)}$$

- Supondremos colecciones no vacías ($N > 0$) y términos clave que aparecen en algún documento ($ni > 0$). Por lo tanto $N/ni > 0$.
- Como se puede ver, el valor de idf_i es inversamente proporcional al número de documentos donde aparece *ki*.

Inverse document frecuency: Ejemplo

- Imaginemos una colección de $N=4$ documentos y un vocabulario de longitud **t**=3, con las siguientes frecuencias por término clave:
  $n1 = 3$ $n2 = 1$ $n3 = 4$
- El valor idf_i calculado será:
  - $idf_1 = \log (N/n1) = \log (4/3) =$ **0,415**
  - $idf_2 = \log (N/n2) = \log (4/1) =$ **2**
  - $idf_3 = \log (N/n3) = \log (4/4) =$ **0**

Inverse document frecuency

- La formulación concreta de la **frecuencia inversa de documento** surge de la intuición y de la experiencia.
- Se ha observado empíricamente que esta formulación da lugar a buenos valores para medir la importancia discriminante.
- Por ello, la frecuencia de término clave se adapta para usar logaritmos, y así poder combinarlos de manera directa.

Pesos TF-IDF

- Una vez que tenemos los dos ingredientes (TF e IDF), el peso w_{ij} (o w_{iq}) de un término clave *ki* y en un documento *dj* (o consulta *q*) utilizando TF-IDF se define como:

  $$\boldsymbol{w_{ij} = tf_{ij} \,.\, idf_i}$$

  o de manera desplegada

  $w_{ij} = (1 + \log f_{ij}).\ \log (N/ni)$, si $f_{ij} > 0$

  $w_{ij} = 0$, si $f_{ij} = 0$

Pesos TF-IDF: Ejemplo

- Consideremos que tenemos las siguientes **frecuencias de término clave** para dj

  $tf_{1j} = \mathbf{3}$ $tf_{2j} = \mathbf{0}$ $tf_{3j} = \mathbf{2}$

  y unas **frecuencias inversas de documento**

  $idf_1 = \mathbf{0{,}415}$ $idf_2 = \mathbf{2}$ $idf_3 = \mathbf{0}$

  entonces los pesos de los términos clave para el documento *dj* son:

  - $w_{1j} = tf_{1j} \,.\, idf_1 = 3 \,.\, 0{,}415 = \mathbf{1{,}245}$
  - $w_{2j} = tf_{2j} \,.\, idf_2 = 0 \,.\, 2 = \mathbf{0}$
  - $w_{3j} = tf_{3j} \,.\, idf_3 = 2 \,.\, 0 = \mathbf{0}$

Pesos TF-IDF

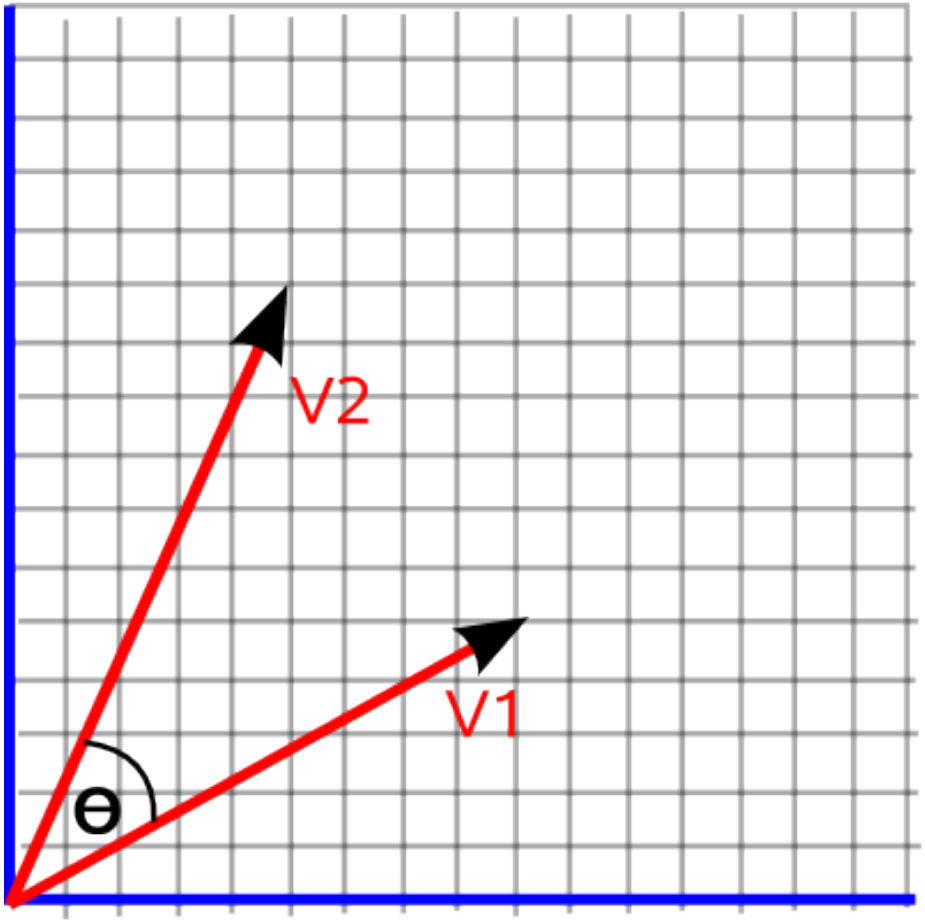
- Las fórmulas para TF e IDF que hemos visto son las clásicas y más utilizadas, aunque no son las únicas opciones.
- Podéis encontrar más variantes en: **Modern Information Retrieval, the concepts and technology behind search**, second edition. *Ricardo Baeza-Yates y Berthier Ribeiro-Neto*. Pearson Education Limited (2011).

Modelo vectorial

- Para asignar pesos a las componentes de un vector *dj* usamos TF-IDF:
  - dj = (w_{1j}, w_{2j}, …, w_{tj})
- Las consultas son texto libre (“mejor coche 1998”). Por simplicidad, supondremos que su vector asociado contiene 0 en los términos que no aparecen y 1 en los que aparecen:
  - *q = (1, 0, …, 1)*

Modelo vectorial: similitud

- La similitud **R(dj,q)** se calculará como el ángulo que forman los vectores del documento *dj* y de la consulta *q*.
- Concretamente utilizaremos el **coseno** del ángulo que forman.



Modelo vectorial: similitud

- En general el coseno toma valores entre [-1, 1], sin embargo nuestro caso es más concreto: todos los pesos son ≥ 0, por lo que los ángulos estarán siempre entre 0º y 90º:
  - El valor del coseno estará entre 1 (máxima similitud, ángulo 0º) y 0 (mínima similitud, ángulo de 90º).

Modelo vectorial: similitud

- ¿Cómo medimos el coseno del ángulo Θ formado por dos vectores *a* y *b*?
- Utilizando la fórmula del **producto escalar**:

  $a \bullet b = |a| \cdot |b| \cdot \cos \Theta$

- Por lo tanto

  $R(a,b) = \cos \Theta = (a \bullet b) / (|a| \cdot |b|)$

Modelo vectorial: similitud

- Considerando los vectores, tenemos que:

$$dj \cdot q = \sum_{i=1}^{t} w_{ij} w_{iq} = (w_{1j} w_{1q}) + (w_{2j} w_{2q}) + \ldots + (w_{tj} w_{tq})$$

$$|dj| = \sqrt{\sum_{i=1}^{t} w_{ij}^2} = \sqrt{w_{1j}^2 + w_{2j}^2 + \ldots + w_{tj}^2}$$

$$|q| = \sqrt{\sum_{i=1}^{t} w_{iq}^2} = \sqrt{w_{1q}^2 + w_{2q}^2 + \ldots + w_{tq}^2}$$

Modelo vectorial: similitud

- En resumen:

$$R(dj,q) = \frac{dj \cdot q}{|dj||q|} = \frac{\sum_{i=1}^{t} w_{ij} w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

Modelo vectorial: ejemplo de similitud

- Consideremos t=3 y los siguientes vectores sencillos para 2 documentos y una consulta
  - d1 = (1.245, 0, 0)
  - d2 = (1.1, 0.78, 0)
  - q = (0, 1, 1)
- Tenemos que
  - R(d1,q) = 0 / |d1|.|q| = **0**
    (son vectores perpendiculares)
  - R(d2,q) = 0,78 / 1,348 . 1,414 = **0,409**

Modelo vectorial: resumen

- Al utilizar pesos para cada término clave, **mejora la calidad** de la recuperación
- Al permitir **encaje parcial**, permite devolver documentos que *aproximan* la consulta (no la satisfacen entera)
- Al usar una medida de similitud en [0,1], permite **ordenar** los documentos obtenidos.

Bibliografía

Bibliografía

- **Information Retrieval: Implementing and Evaluating Searching Engines.** Stefan Bütcher, Charles L. A. Clarke, Gordon V. Cormak. The MIT Press (2016).
- **Modern Information Retrieval, the concepts and technology behind search**, second edition. *Ricardo Baeza-Yates y Berthier Ribeiro-Neto*. Pearson Education Limited (2011).

Bibliografía

- **Introduction to Information Retrieval.** *Christopher D. Manning, Prabhakar Raghavan y Hinrich Schütze*. Cambridge University Press. 2008. http://www-nlp.stanford.edu/IR-book/
- **Search Engines: Information Retrieval in Practice** (International Edition). *W. Bruce Croft, Donald Metzler y Trevor Strohman*. Person Education. 2010.