Recuperación de la Información

Text Document Retrieval

Semana 06

Heider Sanchez - *hsanchez@utec.edu.pe*

Document Retrieval

Ranking Retrieval

Introduction to Information Retrieval, by C. Manning, P. Raghavan, and H. Schütze (Cambridge University Press, 2008).

Problemas con la búsqueda booleana:

- Hasta ahora, todas nuestras consultas han sido booleanas.
  - Los documentos coinciden o no.
- Bueno para usuarios expertos con una comprensión precisa de sus necesidades y la colección.
  - También es bueno para las aplicaciones: las aplicaciones pueden consumir fácilmente miles de resultados.
- Pero no es bueno para la mayoría de los usuarios.
  - La mayoría de los usuarios son incapaces de escribir consultas booleanas (o lo son, pero creen que es demasiado trabajo).
  - La mayoría de los usuarios no quieren pasar a través de miles de resultados.
    - Esto es particularmente cierto en la búsqueda web.

Problemas con la búsqueda booleana: “festival o hambruna”

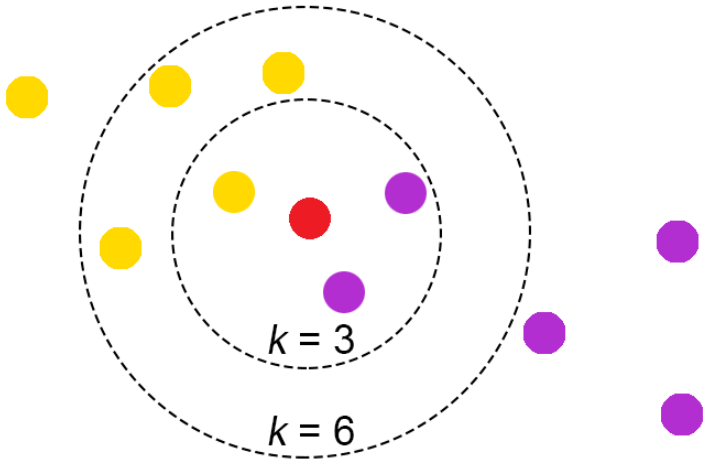
- Las consultas booleanas a menudo dan como resultados muy pocos (= 0) o demasiados (1000).
- Query 1: “*standard user dlink 650*” → 200,000 results
- Query 2: “*standard user dlink 650* ***no*** *card found*”: 0 results
- Se requiere mucha habilidad para llegar a una consulta que produzca un número manejable de resultados.
    - AND da muy pocos; OR da demasiados.

Ranked Retrieval Model

- En lugar de un conjunto de documentos que satisfacen una expresión de consulta, en el **ranked retrieval**, el sistema devuelve un **orden** de documentos (top) de la colección para una consulta dada.
- Consultas de texto libre: en lugar de un lenguaje de consulta de operadores y expresiones, la consulta es solo una o más palabras en lenguaje natural.
- En principio, hay dos opciones separadas aquí, pero en la práctica, el **ranked retrieval** normalmente se ha asociado con consultas de texto libre y viceversa.

Ranked Retrieval: Scoring

- Cuando un sistema produce un set de resultados ranqueados, los sets de resultados grandes no son un problema:
  - De hecho, el tamaño del conjunto de resultados no es un problema
  - Solo mostramos los mejores resultados de k (≈ 10)
  - No abrumamos al usuario
  - Premisa: *el algoritmo de ranking funciona*.



Ranked Retrieval: Scoring

- Scoring la base de la recuperación por ranking.
- Deseamos devolver los documentos en orden para que sean mas útiles al usuario.
- ¿Cómo podemos ranquear los documentos de la colección con respecto a una consulta?
- Asignando un score – entre [0, 1] – a cada documento.
- Este score mide que tan bien “coinciden” el document y la consulta.

Ranked Retrieval: query-document matching scores

- Necesitamos una forma de asignar un score al par (consulta , documento).
- Empecemos con un término de la consulta
    - Si el término de la consulta no ocurre en el documento, el score debería ser 0.
    - Cuanto más frecuente sea el término en el documento, mayor será el score (debería serlo).
- Vamos a ver una serie de alternativas para esto.

Ranked Retrieval: Jaccard coefficient

- Es una medida comúnmente utilizada para medir la superposición entre dos conjuntos A y B.
- jaccard$(A,B) = |A \cap B| \ / \ |A \cup B|$
- jaccard$(A,A) = 1$
- jaccard$(A,B) = 0$ if $A \cap B = 0$
- A y B no tienen el mismo tamaño.
- Siempre se asigna un número entre 0 y 1.

Ranked Retrieval: Jaccard coefficient

- Ejemplo de scoring:
  - ¿Cuál será el score de coincidencia entre consulta y documento que el coeficiente de Jaccard calcula para cada uno de los siguientes documentos ?
  - Query: *ides of march*
  - Document 1: *caesar died in march*
  - Document 2: *the long march*

Ranked Retrieval: Jaccard coefficient

- **Problemas con Jaccard:**
  - No considera la **frecuencia del término** (cuántas veces aparece un término en un documento)
  - Los términos raros en una colección son más informativos que los términos frecuentes. Jaccard no considera esta información.
  - Necesitamos una forma más sofisticada de normalizar la longitud.
  - Una solución rápida sería lo siguiente:

$$|A \cap B| / \sqrt{|A \cup B|}$$

Ranked Retrieval:

(recordar) Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

En donde cada documento es representado por un vector binario $\in \{0,1\}^{|V|}$

Ranked Retrieval: Term-document count matrices

- Considere la cantidad de ocurrencias de un término en un documento:
  - Cada documento es un vector de conteo $\mathbb{N}^{v}$:

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Ranked Retrieval: Term frequency **tf**

- La frecuencia del término $tf_{t,d}$ del termino *t* en un documento *d* es definido como el número de veces que ocurre *t* en *d*.
- Queremos utilizar tf para calcular el score de coincidencia entre documento y consulta. ¿Pero cómo?
- La frecuencia del término en bruto no es lo que queremos:
    - Un documento con 10 apariciones del término sería más relevante que un documento con una sola ocurrencia del término.
    - Pero no es 10 veces más relevante.
- **La relevancia no aumenta proporcionalmente con la frecuencia del término.**

Tener en cuenta: frecuencia = conteo en IR

Ranked Retrieval: Log-frequency weighting

- El log-frequency weight de un término t en d es:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- El score para el par documento-consulta: suma de los términos *t* que coinciden en ambos *q* y *d*:

$$\mathrm{Score}(q,d) = \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$$

- El score es 0 si ninguno de los términos de la consulta está presente en el documento.

Ranked Retrieval: Document frequency

- Los términos raros son más informativos que los términos frecuentes
    - Recordar los stop words
- Considere un término en la consulta que sea raro en la colección (por ejemplo, *aracnocéntrico*)
- Un documento que contiene este término es muy probable que sea relevante para la consulta *aracnocéntrica.*

→ Queremos un peso elevado para términos raros como aracnocéntrico.

Ranked Retrieval: Document frequency

- Los términos frecuentes son menos informativos que los términos raros.
- Considere un término de consulta que sea frecuente en la colección (por ejemplo, alto, aumentar, línea).
- Es más probable que un documento que contenga dicho término sea más relevante que un documento que no lo tenga.
- Pero no es un indicador seguro de relevancia.
- → Para términos frecuentes, queremos pesos altos y positivos, para palabras como alto, aumentar y línea.
- Pero a su vez, sus pesos deben ser más bajos que para los términos raros.
- Usaremos la frecuencia de documentos (df) para capturar esto.

Ranked Retrieval: idf weight

- df_t es la frecuencia de documento de *t*: número de documentos que contienen a *t*
  - df_t es una medida inversa de la informatividad de *t*
  - $df_t \leq N$
- Definimos idf (frecuencia de documento inverso) de *t* mediante:
  - Usamos log (N /dft) en lugar de N/dft para "amortiguar" el efecto de idf.

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

La base del log es irrelevante

Ranked Retrieval: idf weight

- Ejemplo, suponer *N* = 1 millón

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

Hay un solo valor idf para cada término t en una colección.

Ranked Retrieval: Collection vs. Document frequency

- La frecuencia de coleccion de *t* es el número de ocurrencias de *t* en la colección, contando multiples ocurrencias.
- Ejemplo:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- ¿Qué palabra es un mejor término de búsqueda (y debería tener un mayor peso)?

Ranked Retrieval: tf-idf weighting

- El peso tf-idf de un término es el producto de sus pesos tf e idf.

- El mejor esquema de ponderación conocido en recuperación de información:

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

  - Nombres alternativos: tf.idf, tf x idf

- Aumenta con el número de ocurrencias dentro de un documento.
- Aumenta con la rareza del término en la colección.

Ranked Retrieval: tf-idf weighting

- El score para un documento dado una consulta sería:

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

- Hay muchas variantes:
  - Cómo se calcula “tf” (con / sin log)
  - Si los términos en la consulta también están ponderados…

Ranked Retrieval: Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Cada documento está representado por un vector de valores reales de los pesos tf-idf $\in \mathbb{R}^{|V|}$

Ranked Retrieval: documentos como vectores

- Entonces tenemos un espacio vectorial de |V| dimensiones.
- Los terminos son los ejes del espacio
- Los documentos son puntos o vectores en este espacio.
- Alta dimensionalidad: decenas de millones de dimensiones cuando se aplica esto a un motor de búsqueda en la web.
- Estos son vectores muy dispersos, la mayoría de las entradas son cero.

Ranked Retrieval: consultas como vectores

- **Key idea 1:** Haga lo mismo para las consultas: represéntelas como vectores en el espacio.
- **Key idea 2:** ranquear los documentos según su proximidad a la consulta en este espacio.
- Proximidad = similitud de vectores
- Proximidad ≈ inversa de la distancia
- Entonces: ranquear los documentos más relevantes más alto que los documentos menos relevantes.

Ranked Retrieval: Vector space model

- Vector space = all the terms encountered

  $<t_1, \ t_2, \ t_3, \ldots, t_n>$

- Document

  $D = \ < a_1, a_2, a_3, \ldots, a_n>$

  a_i = weight of t_i in D

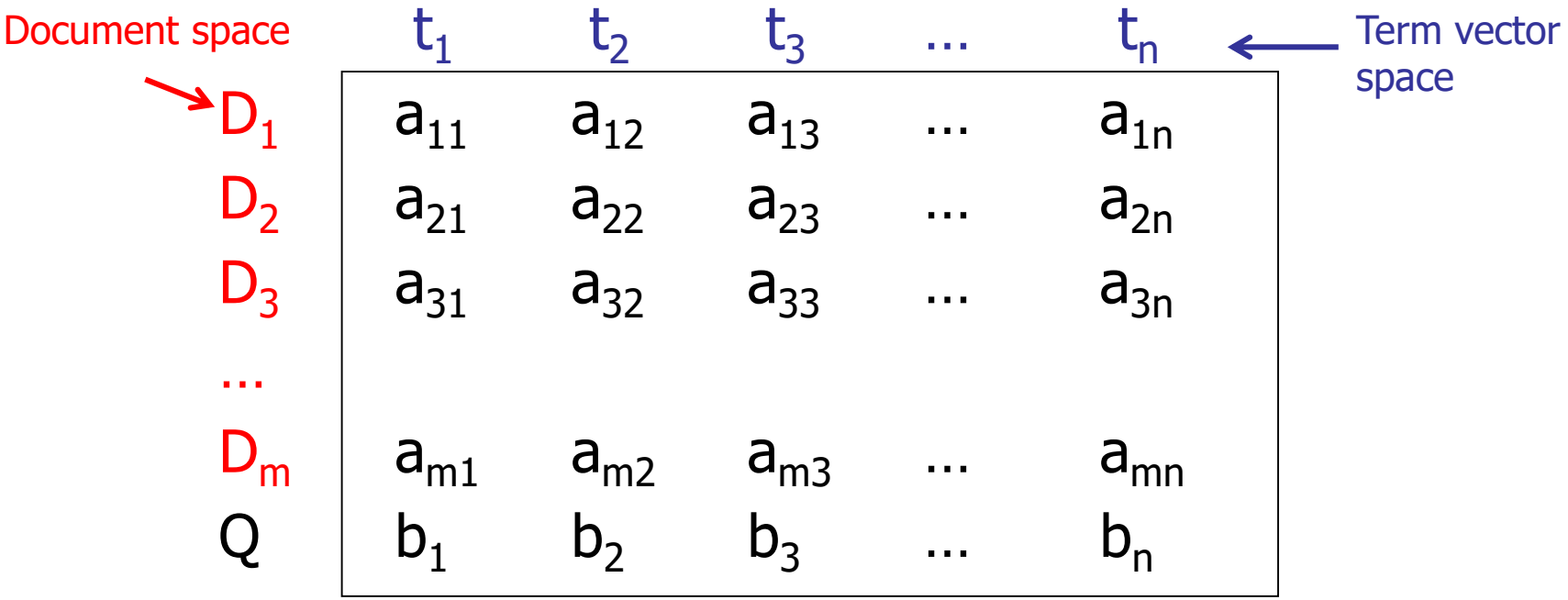
- Query

  $Q = \ < b_1, b_2, b_3, \ldots, b_n>$

  b_i = weight of t_i in Q

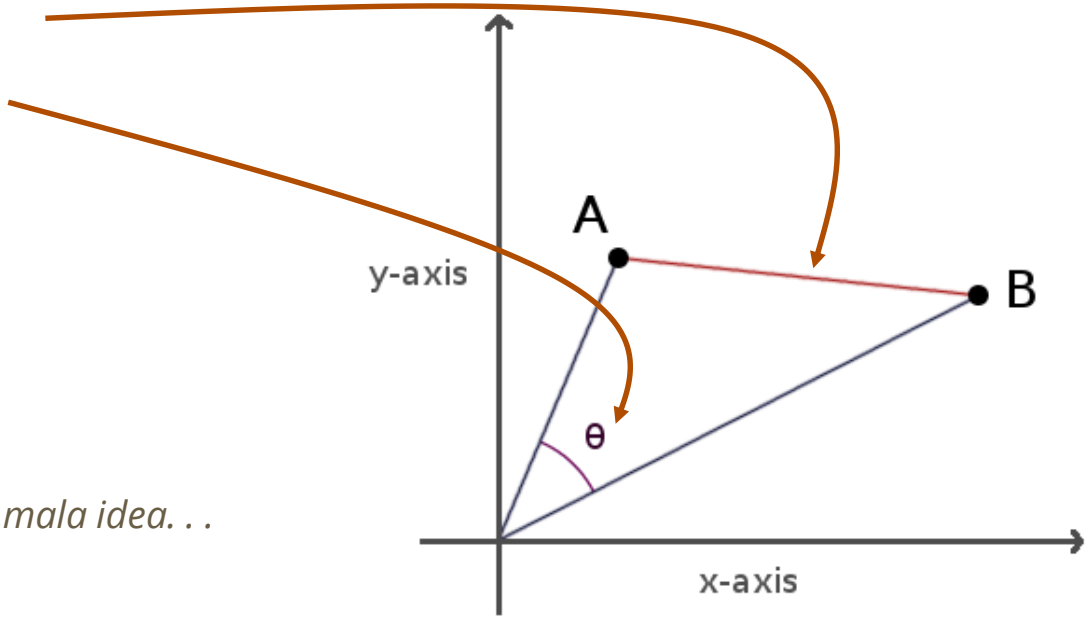
- Score(D,Q) = Sim(D,Q)

Ranked Retrieval: representación de la matriz



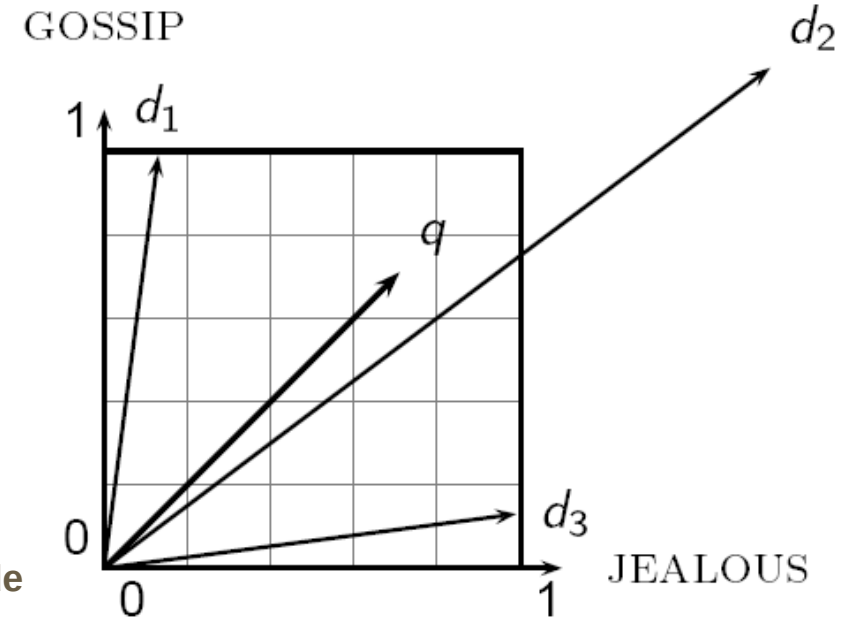
Ranked Retrieval: Proximidad

- Formalización del espacio vectorial de proximidad.
  - Euclidean distance?
  - Cosine distance?
  - *Distancia Euclidiana es una mala idea. . .*



Ranked Retrieval: Proximidad

La distancia Euclediana entre q y d_2 es grande aunque la distribución de términos en la consulta q y la distribución de términos en el documento d_2 son muy similares.



La distancia Euclidiana es grande para vectores de diferentes longitudes.

Ranked Retrieval: Proximidad

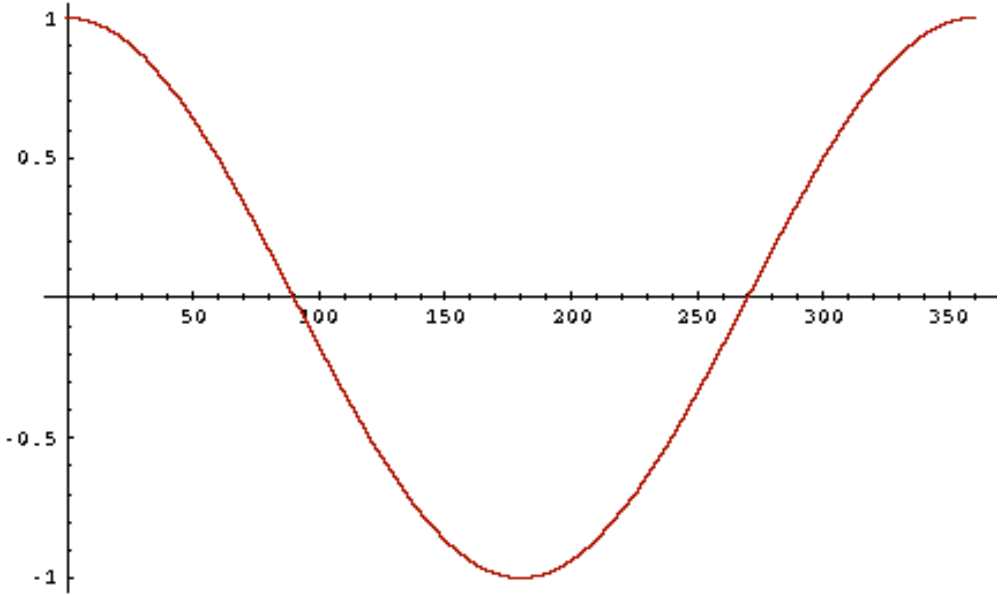
- **Usando el ángulo en lugar de la distancia**
  - Experimento mental: tome un documento d y adjúntelo a sí mismo. Llama a este documento d '.
  - "Semánticamente" d y d ' tienen el mismo contenido.
  - La distancia euclidiana entre los dos documentos puede ser bastante grande.
  - El ángulo entre los dos documentos es 0, el cual corresponde a la similitud máxima.
  - **Key idea: ranquear los documentos según el ángulo que forman con la consulta.**

Ranked Retrieval: Proximidad

- **De ángulos a cosenos**
  - Las dos nociones siguientes son equivalentes:
    - Ranquear los documentos en orden decreciente del ángulo entre la consulta y el documento.
    - Ranquear documentos en orden creciente de coseno (consuta,documento)
  - El coseno es una función monótonamente decreciente para el intervalo [0º, 180º]

Ranked Retrieval: Proximidad

- **De ángulos a cosenos**



- ¿Cómo --y por qué-- debemos calcular los cosenos?

Ranked Retrieval: cosine(query,document)

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

q_i is the tf-idf weight of term i in the query
d_i is the tf-idf weight of term i in the document

$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Ranked Retrieval: cosine(query,document)

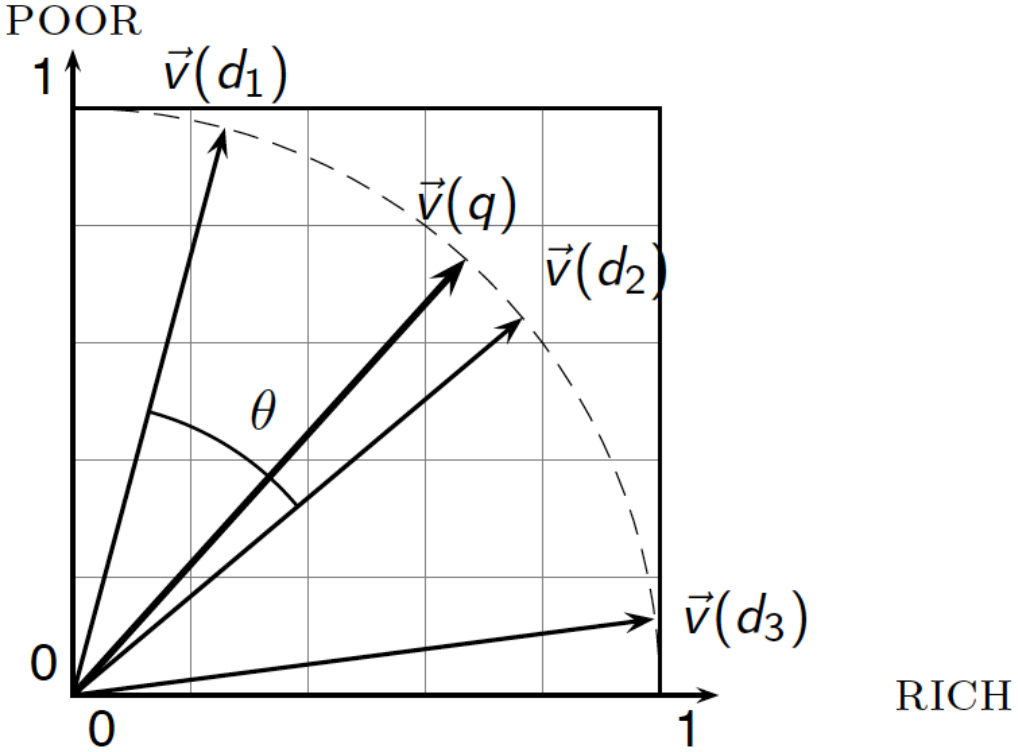
- **Normalizar la longitud:**
  - Dividir un vector por su norma lo convierte en un vector unidad (longitud) (en la superficie de la unidad hiperesférica)

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

  - El efecto en los dos documentos d y d ' (d anexado a sí mismo) de la diapositiva anterior:
    - Tienen vectores idénticos después de la normalización de la longitud.
    - Los documentos largos y cortos ahora tienen pesos comparables.

Ranked Retrieval: cosine(query,document)

- **Ilustración de la similitud coseno:**



Ranked Retrieval: similitud coseno entre tres documentos

Que tan similares son las novelas:

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*

WH: *Wuthering Heights*

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Frecuencia de terminus (conteo)

Nota: para simplificar este ejemplo, no hacemos ponderación idf.

Ranked Retrieval: similitud coseno entre tres documentos

- Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

- After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

cos(SaS,PaP) ≈
0.789 × 0.832 + 0.515 × 0.555 + 0.335 × 0.0 + 0.0 × 0.0
≈ 0.94
cos(SaS,WH) ≈ 0.79
cos(PaP,WH) ≈ 0.69

¿Por qué tenemos cos(SaS, PaP) > cos(SaS, WH)?

Ranked Retrieval: similitud coseno

Computing cosine scores

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5      for each pair(d, tf_{t,d}) in postings list
 6      do Scores[d] += w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

Ranked Retrieval : ejemplo de tf-idf

Document: *car insurance auto insurance*
Query: *best car insurance*

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt	n'lize	tf-raw	tf-wt	wt	n'lize	
auto	0	0	5000	2.3	0	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0.34	0	0	0	0	0
car	1	1	10000	2.0	2.0	0.52	1	1	1	0.52	0.27
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3	0.68	0.53

Exercise: what is *N*, the number of docs?

Doc length = $\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

Score = 0+0+0.27+0.53 = 0.8

Ranked Retrieval: resumen – ranking de espacios vectoriales

- Representar la consulta como un vector de pesos tf-idf.
- Representar cada document como un vector de pesos tf-idf.
- Calcular el score de la similitude coseno para la consulta y cada vector de documento.
- Ranquear los documentos a la consulta con su respectivo score.
- Retornar el Top-K al usuario que hizo la consulta.

Document Retrieval

Ranking Retrieval

Introduction to Information Retrieval, by C. Manning, P. Raghavan, and H. Schütze (Cambridge University Press, 2008).