

# Búsqueda y recuperación de información

*¿Cómo definimos el problema de la búsqueda de información no estructurada?*

*¿Cómo decidir con qué criterio seleccionar y ordenar las respuestas?*

*¿Cómo traducir estos criterios a una fórmula?*

*¿Cómo desarrollar soluciones viables a escala masiva?*

# Búsqueda y recuperación de información

- ♦ Introducción
- ♦ Conceptos fundamentales
- ♦ Modelos de IR
- ♦ Índices de búsqueda
- ♦ Evaluación

# Terminología y conceptos fundamentales

(Para simplificar, usaremos terminología de búsqueda, extrapolable en términos generales a otras aplicaciones de IR)

- ◆ Necesidad de información, consulta
- ◆ Espacio de búsqueda (“colección”)
- ◆ Solución IR
  - Modelo (formulación teórica / algorítmica)
  - Ránking, función de ránking
  - Detalles de implementación: indexación, sistema y componentes
- ◆ Relevancia
- ◆ Eficacia (calidad), eficiencia (tiempo, coste)

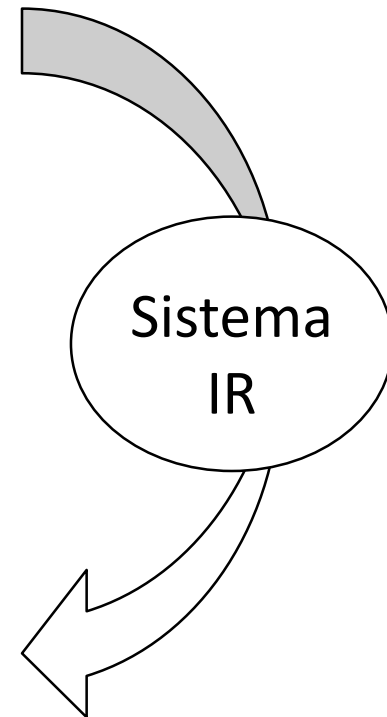
# Recuperación de Información

## 1. Acceso a un espacio de información **no estructurada**



## 2. **Factor humano** en la definición del problema

- Un usuario final decide lo que es o no relevante



# ¿Qué es la recuperación de información?

- ♦ Ayudar al usuario a llegar a una información de interés en espacios masivos de información no estructurada
  - Información **no estructurada**
  - Y/o la búsqueda del usuario no tiene una formulación exacta
  - ⇒ La respuesta sólo puede ser **aproximada**
- ♦ Recuperación de información es, por ejemplo:
  - Encontrar información en la Web a partir de una lista de palabras
  - Responder a una pregunta en lenguaje natural
  - Recomendar compras, vídeo, música, tweets, contactos... a un usuario analizando su histórico de actividad en el sistema
- ♦ Recuperación de información no es, por ejemplo:
  - Consultar a una base de datos
  - Buscar una cadena de caracteres en un documento de texto

Incertidumbre


# “Historia abreviada”

- ♦ Information Retrieval (IR), desde los 50
- ♦ La mayoría de los **fundamentos** básicos se desarrollaron en los 60-70
  - Modelos, algoritmos, metodologías de evaluación
- ♦ Boom de la **Web** en los 90
  - Enorme impulso al campo de IR → tendemos a asociar “IR = buscador Web”  
Es cierto, aunque hay más: búsqueda local, recomendación, QA, resúmenes automáticos, detección de topics, búsqueda multimedia, vertical, geográfica...
- ♦ Muchos cambios desde mediados de los 90 hasta hoy
  - PageRank, técnicas probabilísticas, query logs, machine learning, personalización, recomendación, diversidad, computación masiva, redes sociales...
- ♦ Sigue siendo un área muy dinámica
  - Investigación teórica, ciencia aplicada, ingeniería (academia + industria)

# Entender la naturaleza del problema de la recuperación de información

- ♦ La búsqueda parte de una **necesidad del usuario**
- ♦ La necesidad surge de un **contexto**
  - Tomar una decisión: qué coche comprar, qué carrera estudiar, dónde ir de vacaciones...
  - Ganar un conocimiento: influencia del Neanderthal en el humano moderno, cómo es la vida en Siberia...
  - Gestionar un problema: identificar y tratar una lesión...
  - Realizar una tarea: tengo que implementar un perceptrón...
  - Etc.
- ♦ Para la calidad de la solución es fundamental entender el problema del usuario y el sentido de la búsqueda dentro de él

# Entender la naturaleza del problema de la recuperación de información


- ♦ El acierto de las respuestas es subjetivo
- ♦ El reto es conseguir un acierto subjetivo con soluciones automáticas
  - Primer nivel: modelos matemáticos generalistas “asépticos” 
  - Segundo nivel: gran cantidad de heurísticas y ajustes particulares para adecuarse al comportamiento humano



# Imprecisión, informalidad, incertidumbre

- ♦ En la necesidad de información
  - “¿Las playas de Kenya merecen la pena?”
- ♦ En la formulación de una consulta
  - “kenya beach”, “kenya coast tourism”, “nice kenya beach”, “best keyna beach”, “what are kenya beaches like”...
- ♦ En la información contenida en el espacio de respuestas (documentos)
  - El lenguaje natural...
- ♦ En la respuesta del sistema
  - Siempre aproximada

# Definición del problema

- ♦ Devolver al usuario la mayor cantidad posible de **información relevante** para su **necesidad de información**  
**¿Qué es?**
- ♦ Y la menor cantidad posible de información no relevante

# Conceptos importantes

- ◆ Necesidad de información
- ◆ Colección
- ◆ Solución IR
  - Modelo (formulación teórica / algorítmica)
  - Ránking (resultado práctico)
  - Detalles de implementación: indexación
- ◆ Relevancia
- ◆ Evaluación

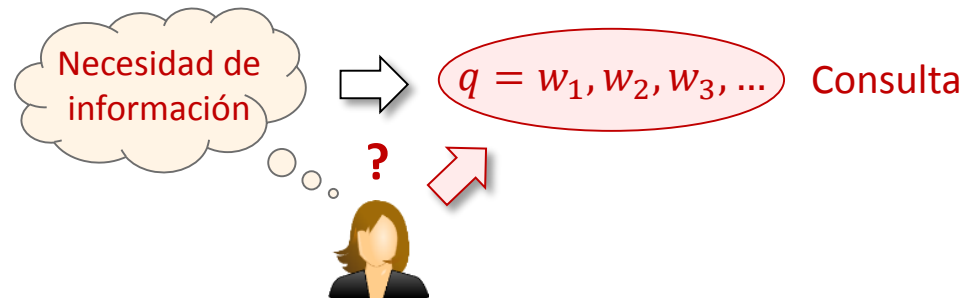
# Necesidad de información

Distintos tipos de necesidad de información motivan soluciones tecnológicas muy distintas

- ◆ Consulta estructurada a base de datos  
(directa en SQL o indirecta vía formulario)
- ◆ Consulta a motor de búsqueda
- ◆ Consulta a sistema de question answering
- ◆ Sistema de recomendación
- ◆ ...

# Necesidad de información

- ♦ Una información de la que el usuario precisa para realizar un objetivo
  - Puede ser muy precisa y clara o muy vaga, incluso (parcialmente) inconsciente
  - El usuario puede tener incertidumbre respecto a cuál es realmente su necesidad
  - Y es dinámica: puede variar considerablemente a lo largo de una sesión de búsqueda
- ♦ El usuario formula su necesidad en forma de consulta al sistema de IR
  - La formulación típica es una lista de palabras, pero hay otras...
  - ...formularios de búsqueda por campos
  - ...preguntas en lenguaje natural (question answering)
  - ...un documento
  - ...o incluso nada: el sistema IR se forma una idea de la necesidad del usuario implícita en las acciones de éste
  - Y no toda necesidad de información requiere un sistema IR!
- ♦ Entre la necesidad de información y la consulta hay un “salto”
  - Hay consultas mejores y peores para una misma necesidad (relevance feedback)
  - Capacidad y habilidad del usuario para traducir necesidades en consultas
  - Expresividad del lenguaje de consulta del sistema



# Espacio de búsqueda

- ♦ Se le suele llamar “colección”
- ♦ Típicamente masivo
- ♦ Comúnmente formado por unidades que llamamos “documentos”
- ♦ El contenido de los documentos es no (o semi) estructurado
  - En esta asignatura: texto
- ♦ Ejemplo: la WWW, la web de una empresa, mi disco duro, mi buzón de correo, Twitter, etc.

# Respondiendo consultas: ránking

- ♦ ¿Devolver el conjunto de respuestas acertadas a una consulta?
- ♦ Pero...
  - Incertidumbre del sistema en cuanto al acierto de cada documento
  - Además, dentro del acierto real, hay grados
  - Y demasiadas respuestas acertadas (p.e. en la Web típicamente varios millones) para que el usuario las vea todas!
- ♦ Por todo ello, una forma adecuada de respuesta es un ránking de respuestas por probabilidad/grado estimado de acierto
  - Una función de ránking que devuelve un valor numérico para cada documento dada una consulta

# Objetivo final: relevancia

- ♦ Un concepto central pero escurridizo
  - La base de la evaluación de sistemas IR
  - La base de algunos modelos formales probabilísticos
- ♦ Una propiedad de un par consulta / documento
  - El documento es relevante si **satisface la necesidad de información** que motiva la consulta
  - Comúnmente considerada binaria (relevante / no relevante)
  - Pero modelos más recientes consideran grados de relevancia
- ♦ Simplificaciones habituales
  - Unidimensionalidad (relevancia temática) → rankings combinados
  - Estabilidad (independencia del tiempo) → modelos temporales
  - Consistencia (independencia de usuario y contexto) → personalización
  - Independencia respecto a documentos ya vistos → diversidad



# Respondiendo consultas: función de ránking

Valor de recuperación, “score”

$$f(q, d_1) \in \mathbb{R}$$

$f(d, q)$  estima en qué medida  $d$  es una buena respuesta a  $q$

$f$

Función de ránking

$q = w_1, w_2, w_3, \dots$

Consulta

Espacio de búsqueda

$d_1$

$d_2$

$d_3$

$\dots$

Relevancia

Necesidad de  
información



Usuario  
final

Ránking

$f(q, d_3)$



$d_3$

$\vee$

$f(q, d_1)$



$d_1$

$\vee$

$f(q, d_2)$



$d_2$

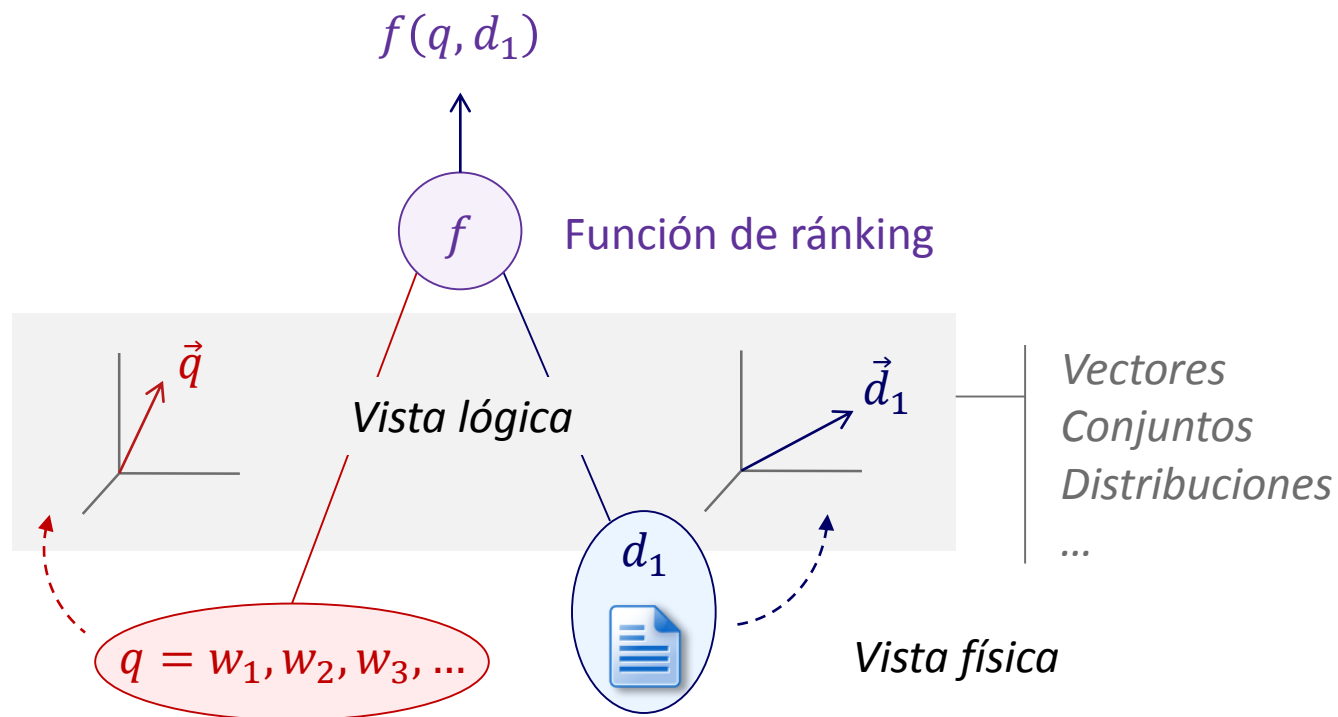
$\vee$

$\vdots$

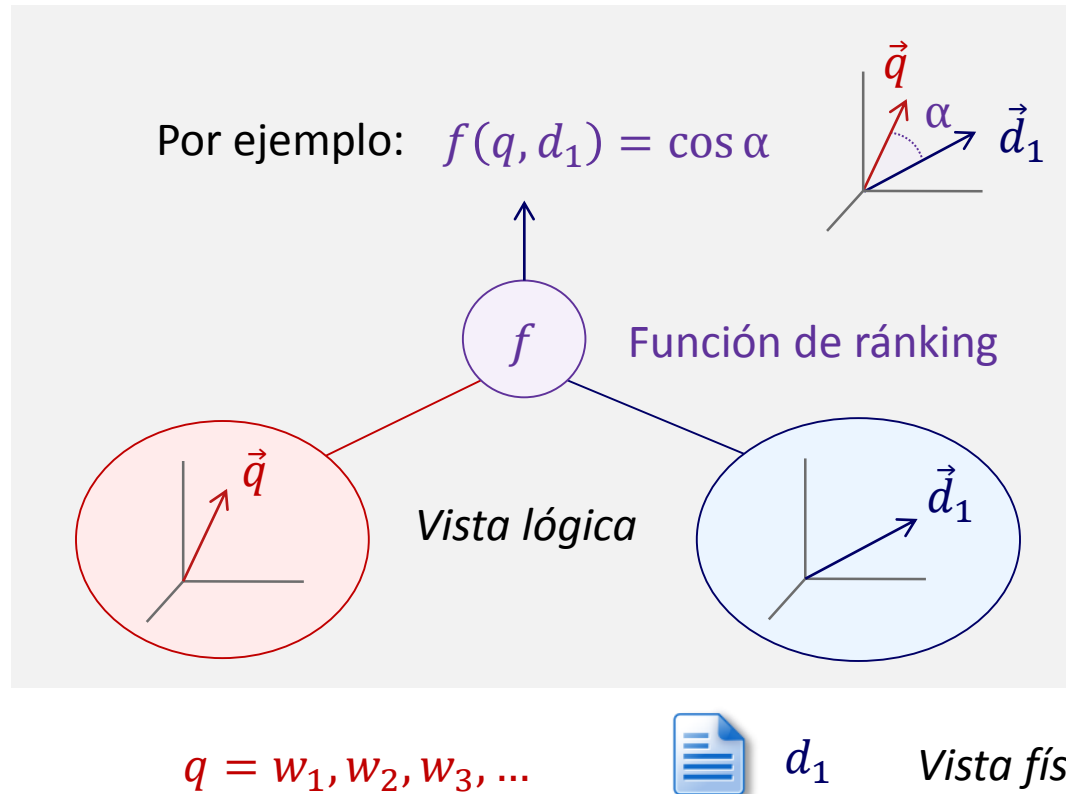
$\vdots$

$\vdots$

# Respondiendo consultas: vista lógica



# Respondiendo consultas: modelo

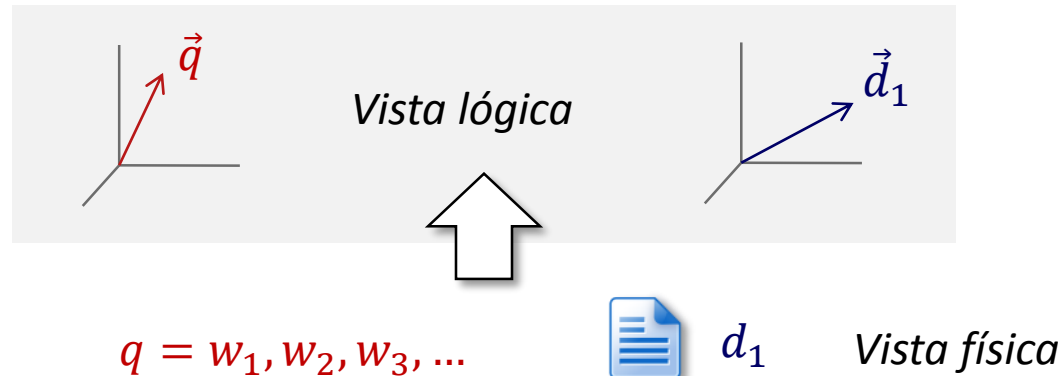


Un **modelo IR** es:

- ♦ Una **representación lógica** de documentos y consultas
  - Incluyendo los métodos y cálculos para construirla
- ♦ Una **función de ránking**  $f(q, d)$  sobre dicha representación
  - Puede ser muy elaborada de calcular

Induce  
el algoritmo  
de ránking

# Respondiendo consultas: indexación



Construir la representación lógica de un espacio de gran escala es altamente costoso

- ♦ Se necesita **construir y actualizar offline**
- ♦ Y almacenar en una estructura eficiente que permita acceso rápido y concurrente en tiempo de consulta → **Índices de búsqueda**

# Indexación

- ♦ Puente entre vista física y lógica
- ♦ Extracción de palabras clave
  - Tokenización, decodificación, eliminación de marcas...
- ♦ Filtrado y procesamiento de las palabras → términos (vocabulario)
  - Stopwords, normalización, stemming, grupos nominales...
- ♦ Construcción de índices optimizados
  - Estructuras, compresión, índices distribuidos
- ♦ Lo veremos más adelante...

# Score vs. relevancia

No confundir relevancia real con estimación de relevancia por un sistema de IR (función de ránking)

- ♦ Juicios de relevancia (ground truth)
  - $d$  es o no relevante para  $q$ ,  $r(d, q) \rightarrow \{0,1\}$
  - $d$  es relevante para  $q$  en mayor o menor grado  $r(d, q) \rightarrow \{0,1,2, \dots, n\}$
- ♦ Función de ránking de un motor de búsqueda
  - $f(d, q) \rightarrow \mathbb{R}$

# Modelos de IR para texto

- ♦ Casi todos están basados en *bag of words*
- ♦ Un formalismo (matemático)
  - **Representación lógica**: un documento es un vector, un conjunto, un evento probabilístico, un nodo de un grafo...
  - **Principios y fundamentos** que rigen las relaciones entre palabras, documentos y consultas
- ♦ Que da lugar a un algoritmo de ránking
  - Una **función de recuperación**  $f: \mathcal{D} \times \mathcal{Q} \rightarrow \mathbb{R}$  que define el ránking
- ♦ Típicamente cada modelo incluye una u otra forma de definir un **peso** para cada palabra en cada documento
  - Se puede ver como una matriz término / documento
  - También se puede ver como términos  $\equiv$  características

# Solución IR



Usuario final

- Necesidad de información
- Formulación de consulta

*Consultas*

*Resultados*

Interfaz de usuario

- Procesamiento de consulta
- Presentación de resultados
- Captura de clicks

Modelo / algoritmia

- Selección de documentos
- Cómputo del ranking

Online

Offline

Indexación

- Extracción de palabras
- Cómputo de pesos
- Construcción de estructuras de índice
- Crawling (en su caso)



Espacio de búsqueda (colección)



# Modelos de IR

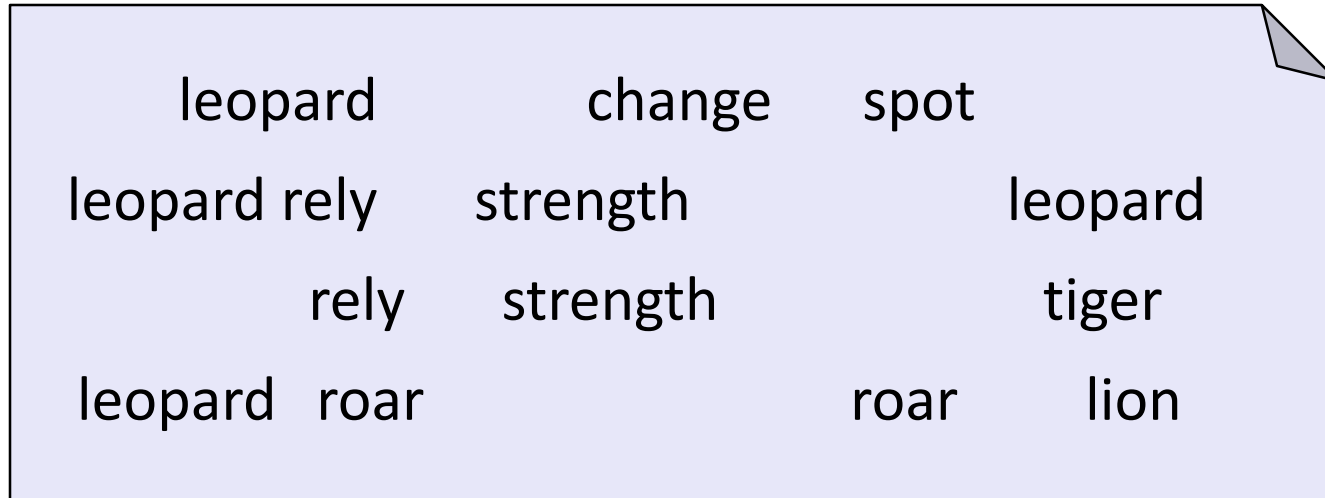
*¿Cómo se determina qué documentos son probablemente relevantes para una consulta?*

# Modelizando documentos

The leopard cannot change its spots. Does the leopard rely on strength? Well no, a leopard does not rely on strength as does the tiger. Leopards roar but not like the roar of a lion.

# Saco de palabras

- ♦ Los modelos fundamentales de IR de texto simplemente extraen y **cuentan las palabras**
- ♦ Y construyen diferentes elaboraciones sobre el “saco” de palabras



leopard change spot  
leopard rely strength leopard  
rely strength tiger  
leopard roar roar lion

# Saco de palabras

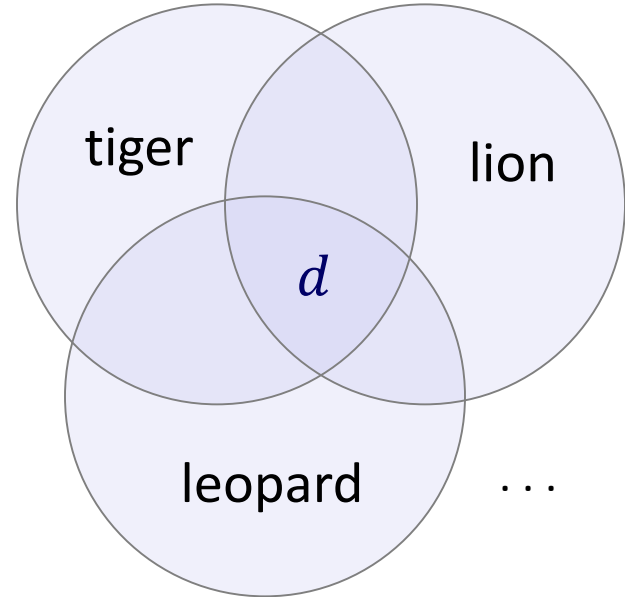
- ♦ Los modelos fundamentales de IR de texto simplemente extraen y **cuentan las palabras**
- ♦ Y construyen diferentes elaboraciones sobre el “saco” de palabras

change	1
leopard	4
lion	1
rely	2
roar	2
spot	1
strength	2
tiger	1

# Modelo booleano

- ♦ Las palabras se ven como **conjuntos de documentos** (los que contienen el término)
- ♦ Las consultas son **operaciones booleanas** ( $\cup$ ,  $\cap$ , complemento) sobre palabras
- ♦ Un documento satisface la consulta si “**pertenece**” a ella

change	1	→	1
leopard	4	→	1
lion	1	→	1
rely	2	→	1
roar	2	→	1
spot	1	→	1
strength	2	→	1
tiger	1	→	1
...		→	0

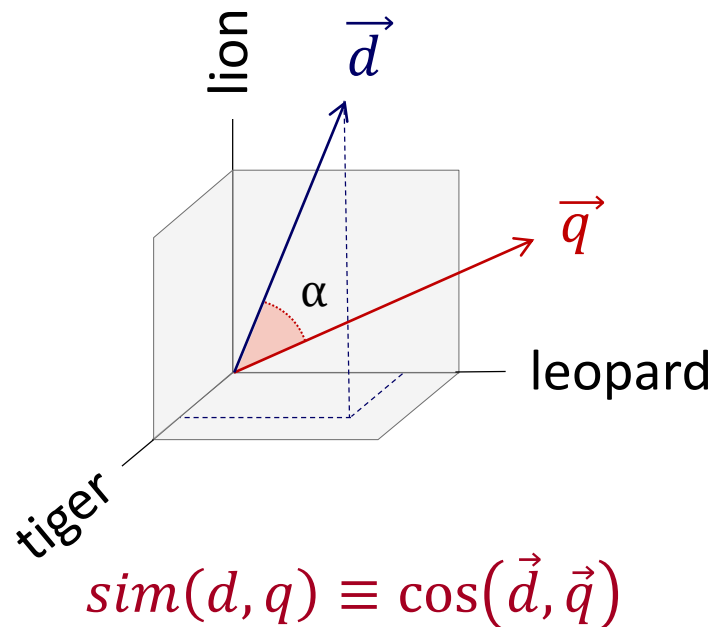


$$\text{sim}(d, q) \equiv [d \in q]$$

# Modelo vectorial

- ♦ Las palabras son ejes de un **espacio vectorial**
- ♦ Los documentos y consultas son vectores en este espacio
- ♦ Un documento satisface la consulta  $\equiv$  el **ángulo** entre los vectores es pequeño

change	1	→	4.3
leopard	4	→	26.9
lion	1	→	9.4
rely	2	→	13.3
roar	2	→	15.3
spot	1	→	6.1
strength	2	→	9.3
tiger	1	→	9.0
...		→	0

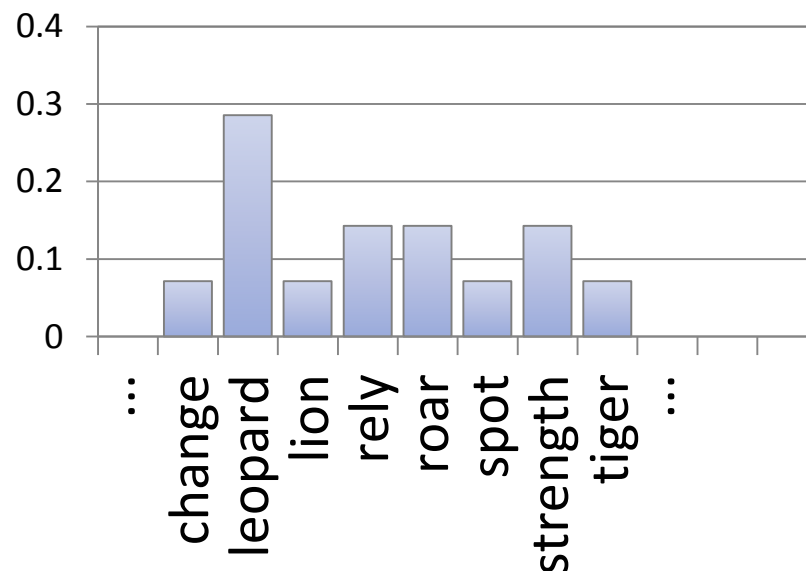


$$tf-idf(\text{leopard}, d) = (1 + \log_2 4) \log_2 \left( \frac{\# \text{ docs in collection}}{\# \text{ docs containing "leopard"}} \right)$$

# Modelos probabilísticos

- ♦ Las palabras son **variables aleatorias**
- ♦ Los documentos y consultas “son” **distribuciones de palabras**
- ♦ Diferentes modelos para valorar la relación entre consultas y documentos



change	1	→	1/14
leopard	4	→	4/14
lion	1	→	1/14
rely	2	→	2/14
roar	2	→	2/14
spot	1	→	1/14
strength	2	→	1/14
tiger	1	→	1/14
...		→	0



$$P(\text{leopard}|d) \sim \frac{\text{\# occurrences of "leopard" in } d}{\text{\# words in } d}$$

$$\text{sim}(d, q) \equiv p(q|d)$$

# Modelos de IR en texto

- ◆ Booleano 
- ◆ Vectorial 
- ◆ Probabilísticos (PRP, LM, DFR...)
- ◆ Factores latentes (SVD, pLSA, LDA)
- ◆ Learning to rank: machine learning aplicada a IR



# Modelo booleano

- ♦ Es el más simple
  - Devuelve los documentos que contienen las palabras de consulta
  - Se pueden articular partículas lógicas: and, or, not
- ♦ Para búsqueda a gran escala no es una solución buena
  - No define ránking, sólo un filtro binario
- ♦ Sin embargo es útil en varios aspectos
  - Para muchos escenarios es suficiente y adecuado: búsqueda en escritorio, en buzón de correo...
  - En búsqueda a gran escala sirve para un primer filtrado sobre el que se sigue otro método (ya sobre un conjunto más reducido)
  - Introduce las partículas lógicas que otros modelos no contemplan

# Modelo booleano

- ♦ Los pesos de los términos son binarios: 1 si aparecen y 0 en otro caso
  - Se ignora la frecuencia de aparición
  - Los documentos se modelizan por tanto como **conjuntos** de términos
  - Las respuestas son **exactas** (tal como se ha definido la tarea)
- ♦ Se puede utilizar **and, or y not** en las consultas
- ♦ Se devuelven los documentos que cumplen la condición expresada en la consulta
  - Poner la consulta en forma normal disyuntiva
  - Representar las componentes de la consulta como vectores binarios de términos (omitiendo los términos no presentes en la consulta)
  - Formar la unión de los documentos que cumplen cada componente conjuntiva
  - Optimización del coste de ejecución (memoria y proceso) sobre el índice: en la unión de intersecciones se ejecutan primero las condiciones *and*, más restrictivas

# Modelo booleano: función de ránking

Dada  $q = q_1 \vee q_2 \vee \cdots \vee q_n$

Donde  $q_i$  es un vector binario de términos

Podemos representar la función de ranking como:

$$f(d, q) = \begin{cases} 1 & \text{si } \exists i : d = q_i \\ 0 & \text{en otro caso} \end{cases}$$

# Ejemplo

$$q = \text{gold} \wedge (\text{silver} \vee \neg \text{truck})$$

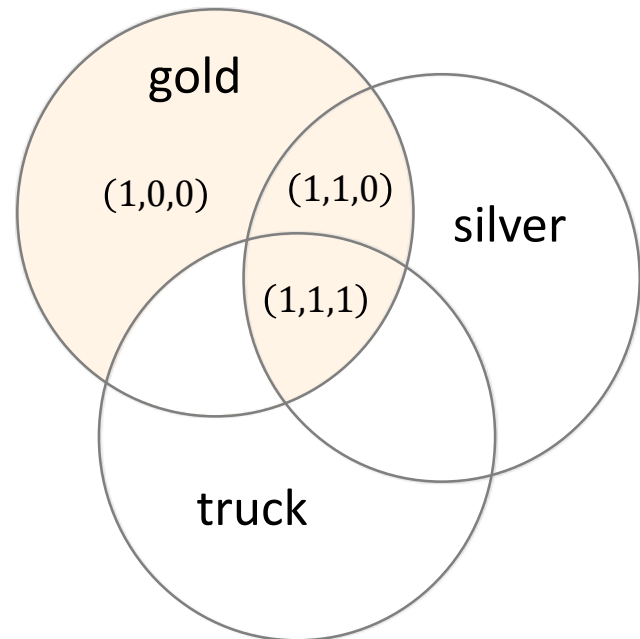
$$d_1 = \text{"Shipment of gold damaged in a fire"} \rightarrow (1,0,0)$$

$$d_2 = \text{"Delivery of silver arrived in a silver truck"} \rightarrow (0,1,1)$$

$$d_3 = \text{"Shipment of gold arrived in a truck"} \rightarrow (1,0,1)$$

$$q = (1,1,0) \vee (1,1,1) \vee (1,0,0)$$

$\downarrow$   
 $d_1$



# Modelo booleano

- ♦ Alternativamente, se pueden representar los términos como vectores binarios de documentos
- ♦ Y las consultas se resuelven operando los vectores coordenada a coordenada (no es necesario pasar la consulta a fnd)

# Ejemplo

$$q = \text{gold} \wedge (\text{silver} \vee \neg \text{truck})$$

$$d_1 = \text{"Shipment of gold damaged in a fire"}$$

$$d_2 = \text{"Delivery of silver arrived in a silver truck"}$$

$$d_3 = \text{"Shipment of gold arrived in a truck"}$$

$$\left. \begin{array}{l} \text{gold} \rightarrow (1,0,1) \\ \text{silver} \rightarrow (0,1,0) \\ \text{truck} \rightarrow (0,1,1) \end{array} \right\} q = (1,0,1) \wedge ((0,1,0) \vee \neg(0,1,1)) = (1,0,0)$$

$\downarrow$   
 $d_1$

# Modelo booleano

En sí mismo, un modelo muy limitado

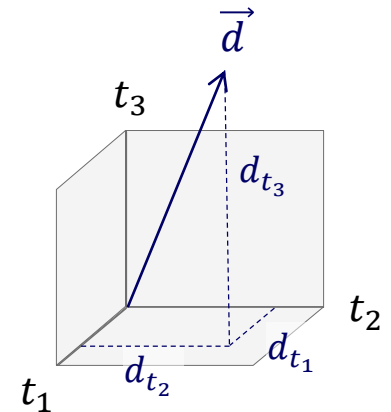
- ♦ **No hay ránking**  $\Rightarrow$  no escala bien
  - Devuelve demasiados documentos o demasiado pocos
- ♦ Las consultas booleanas resultan aparatosas a los usuarios
- ♦ Sin embargo fue el modelo primario durante tres décadas
- ♦ Se sigue usando en funcionalidades de búsqueda sencilla (email, escritorio, bibliotecas...) y como primer filtro de un segundo algoritmo
- ♦ Mejores soluciones: tener en cuenta la **frecuencia** de los términos

# Modelo vectorial (VSM)



Gerard Salton  
(1927-1995)

- ♦ G. Salton, Harvard/Cornell University, 60-70's
- ♦ Se representan documentos y consultas en un espacio vectorial  $\mathbb{R}^{|\mathcal{V}|}$ , donde  $\mathcal{V}$  es el vocabulario
- ♦ La coordenada de los vectores de documento para cada  $t \in \mathcal{V}$  son pesos  $d_t = w(t, \vec{d})$  que se calculan con una fórmula basada en frecuencias
- ♦ ¿Cómo definir una ponderación representativa?
  - Que por un lado cuantifique cuán representativo es cada término en el documento
  - Que por otro matice entre términos muy comunes y otros más específicos (y por tanto significativos)



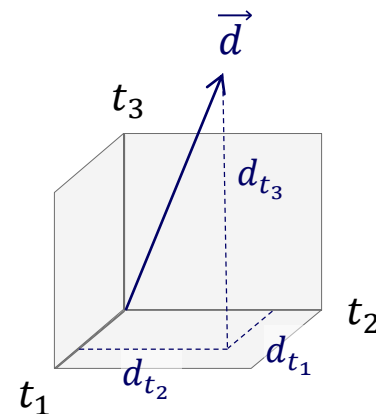


# Modelo vectorial: esquema *tf-idf*

- ♦ El esquema típico de ponderación es *tf-idf*

$$d_t = tf(t, d) \cdot idf(t)$$

- *tf* mide la “importancia” de los términos en los documentos
- *idf* mide el poder de discriminación del término
- ♦ Idea: basar las funciones *tf* e *idf* en la frecuencia de los términos
- ♦ Existen diversas variantes para concretar las funciones *tf* e *idf*, en todas ellas:
  - $tf(t, d)$  es creciente respecto a la frecuencia de  $t$  en  $d$
  - $idf(t)$  mide la especificidad de  $t$  por su frecuencia en la colección



# El esquema *tf-idf*

$$tf(t, d) = \begin{cases} 1 + \log_2 frec(t, d) & \text{si } frec(t, d) > 0 \\ 0 & \text{en otro caso} \end{cases}$$

$$idf(t) = \log \frac{|\mathcal{D}|}{|\mathcal{D}_t|}$$

$\mathcal{D}$  = la colección de documentos  
(espacio de búsqueda)

$\mathcal{D}_t$  = documentos que contienen  
el término  $t$

- ♦  $tf$  tiene que ver con la probabilidad del término en el documento
- ♦ E  $idf$  con la probabilidad en la colección

# El esquema *tf-idf* (cont)

Otras variantes:

$$tf(t, d) = \frac{frec(t, d)}{\max_{t' \in \mathcal{V}} frec(t', d)}$$

- ♦ Pro: evita ventaja para documentos largos
- ♦ Contra: sensible a outliers

$$tf(t, d) = \lambda + (1 - \lambda) \frac{frec(t, d)}{\max_{t' \in \mathcal{V}} frec(t', d)} \quad \text{p. e. } \lambda = 0.5$$

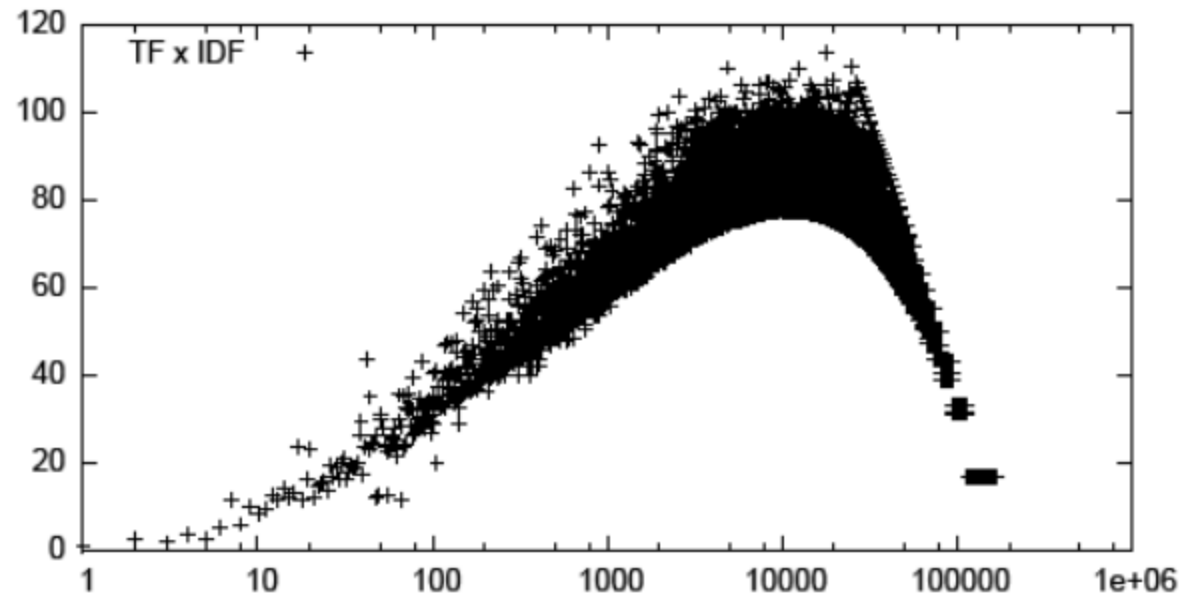
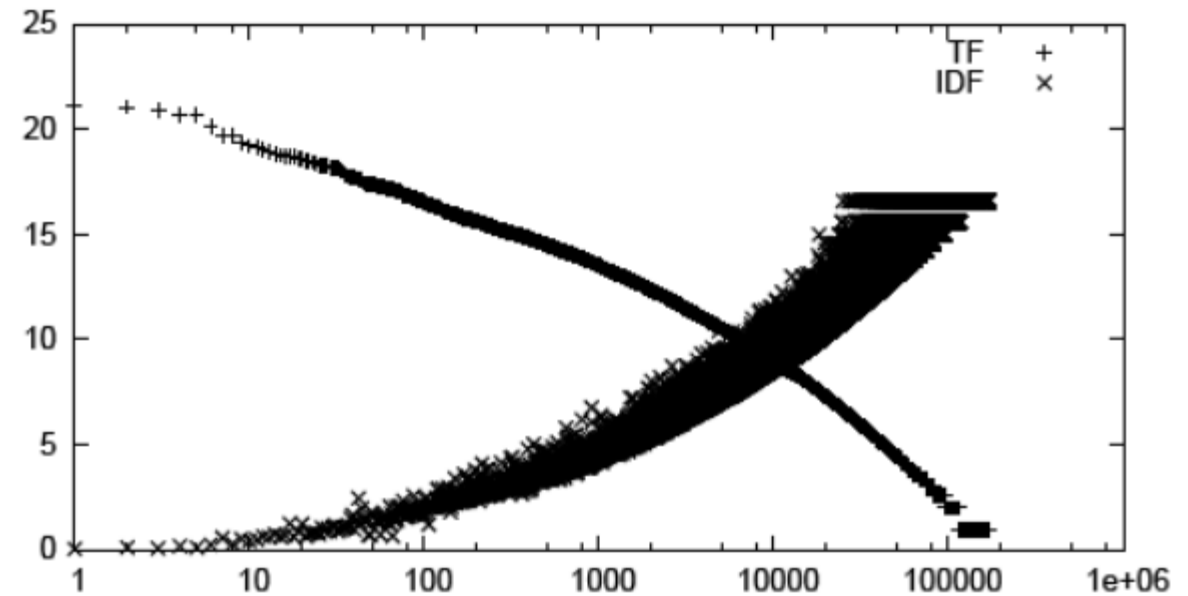
$$idf(t) = \log \frac{|\mathcal{D}| + 1}{|\mathcal{D}_t| + 0.5}$$

...y unas cuantas más (tuning)

# *tf vs. idf*

Plots colección Wall Street Journal, eje  $x$  términos ordenados por  $tf$

- ♦ Comportamiento power law
- ♦  $tf$  e  $idf$  se contrarrestan
- ♦  $idf$  intermedios son los más interesantes



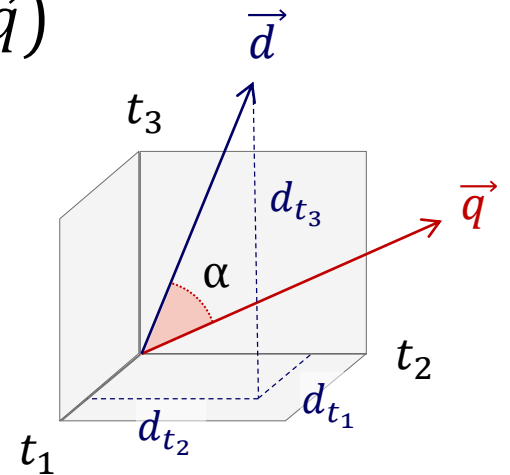
# Modelo vectorial: función de ránking

Finalmente...

- ♦ Construimos  $\vec{q}$ 
  - También por *tf-idf*, aunque no necesariamente con la misma variante
- ♦ Comparamos los vectores  $\vec{d}$  y  $\vec{q}$  en similitud por ángulo

$$f(d, q) = \text{sim}(d, q) = \text{angulo}(\vec{d}, \vec{q}) \propto \cos(\vec{d}, \vec{q})$$

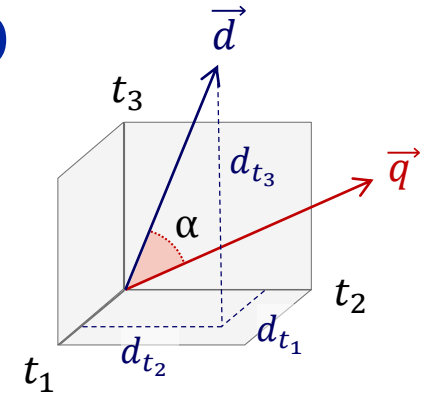
$$\cos(\vec{d}, \vec{q}) = \frac{\vec{d} \cdot \vec{q}}{|\vec{d}| |\vec{q}|} = \frac{\sum_t d_t q_t}{\sqrt{\sum_t d_t^2} \sqrt{\sum_t q_t^2}} \in [0, 1]$$



# Modelo vectorial por coseno

Vector de consulta  $\vec{q}$

- ♦ Se podría hacer  $tf$  binario
  - Salvo que se considere significativa la repetición de términos
  - P.e. aplicaciones donde la consulta es un documento
- ♦  $idf$  penaliza doblemente los términos muy comunes (se puede omitir)
- ♦ Se puede omitir  $|\vec{q}|$  en el denominador



Normalización longitud de documento  $\vec{d}$

- ♦ El módulo del documento  $|\vec{d}|$  en el denominador representa una normalización para evitar el sesgo a documentos largos
- ♦ Se puede normalizar por otras funciones de longitud
  - Tamaño en bytes
  - Nº de palabras
  - Pivoted normalization
  - ...

¿Sería también adecuada la distancia euclídea como alternativa al coseno?

# Otras funciones de similitud

$$jaccard(d, q) = \frac{|d \cap q|}{|d \cup q|} = \frac{|d \cap q|}{|d| + |q| - |d \cap q|}$$

$$dice(d, q) = \frac{2|d \cap q|}{|d| + |q|}$$

- ♦ Jaccard y Dice ignoran las frecuencias
- ♦ No se suelen utilizar tanto en búsqueda de texto pero sí en otras tareas (p.e. recomendación)
- ♦ Las funciones de similitud (incluido el coseno) se pueden utilizar también para comparar documentos ente sí

# Ejemplo

$q$  = “gold silver truck”

$d_1$  = “Shipment of gold damaged in a fire”

$d_2$  = “Delivery of silver arrived in a silver truck”

$d_3$  = “A shipment of gold arrived”

$d_4$  = “The silver was delivered”



# Ejemplo 2

$$1 + \log_2 \text{frec}(t, d)$$



$$\log \frac{|\mathcal{D}|}{|\mathcal{D}_t|}$$



$\text{frec}(t, d)$

$\text{tf}(t, d)$

$\text{tf-idf}(t, d)$

	$d_1$	$d_2$	$d_3$	$d_4$
hielo			4	1
hierba	1	4	2	1
hockey	4			
liga		4	2	
street			1	1
tenis	4		1	

	$d_1$	$d_2$	$d_3$	$d_4$
hielo	0	0	3	1
hierba	1	3	2	1
hockey	3	0	0	0
liga	0	3	2	0
street	0	0	1	1
tenis	3	0	1	0

$\text{idf}(t)$
1
0
2
1
1
1

	$d_1$	$d_2$	$d_3$	$d_4$
hielo	0	0	3	1
hierba	0	0	0	0
hockey	6	0	0	0
liga	0	3	2	0
street	0	0	1	1
tenis	3	0	1	0

$q$
1
1
1

$q$  = "liga street hockey"

$$|d| \begin{bmatrix} \sqrt{45} & 3 & \sqrt{15} & \sqrt{2} \end{bmatrix} \quad \begin{bmatrix} \sqrt{3} \end{bmatrix} |q|$$

$$\frac{d \cdot q}{|d||q|} \longleftarrow \cos(d, q) \begin{bmatrix} 0.52 & 0.58 & 0.45 & 0.41 \end{bmatrix}$$

# Modelo vectorial

- ♦ Primeras versiones de los años 50
  - Sigue siendo muy utilizado hoy día
- ♦ También se utiliza en clustering, clasificación, y otras aplicaciones con documentos de texto
  - También en espacios donde las coordenadas son otro tipo de características (tags, etc.)
- ♦ Aproximación algebraica a la estimación de relevancia
- ♦ Similitud gradual a la consulta, mejora la calidad del ránking
  - Un documento puede ser recuperado aunque no contenga todos los términos de la consulta
- ♦ Incorpora normalización por longitud de forma natural

# Otros modelos

- ◆ Probabilísticos

- PRP, BM25:  $p(R|d, q)$
  - Query likelihood:  $p(q|d)$
  - Document likelihood:  $p(d|q)$
  - KL divergence:  $KL(p(w|q)||p(w|d))$
  - Divergence from randomness
- } “Modelos de lenguaje”

- ◆ Basados en conjuntos

- Vectorial por subconjuntos de términos de la consulta

- ◆ Booleano extendido

- Versiones borrosas de AND y OR

- ◆ Basados en conjuntos difusos

- Generalización del modelo booleano basado en correlación de términos

- ◆ Modelo vectorial generalizado

- Contempla la posibilidad de que los términos no sean “ortogonales”

- ◆ Redes bayesianas

- ◆ ...

# Software open source

- ◆ Lucene

- Modelos IR, crawler (Nutch), HTML parser (Tika), UI (Luke)
- Orientado a aplicaciones

- ◆ Lemur

- Modelos IR, crawler
- Orientado a investigación

- ◆ Terrier

- Modelos IR, crawler
- Orientado a investigación, el más competitivo actualmente

- ◆ ...

- Ver p.e.  
[http://en.wikipedia.org/wiki/List\\_of\\_search\\_engines#Open\\_source\\_search\\_engines](http://en.wikipedia.org/wiki/List_of_search_engines#Open_source_search_engines)

# Elementos principales de Lucene

## ♦ Creación de índice

- **Document**.add(new TextField("content", "..."))
- **IndexWriter**.addDocument(Document)

## ♦ Búsqueda sobre índice

- QueryParser.parse(...) → **Query**
- **IndexSearcher**.search(Query, n).scoreDocs  
→ **ScoreDoc**[n]
- ScoreDoc.doc → docID  
ScoreDoc.score → double