TEMA 6. BÚSQUEDA EN LA WEB

Contenidos basados en el material del curso de Manning

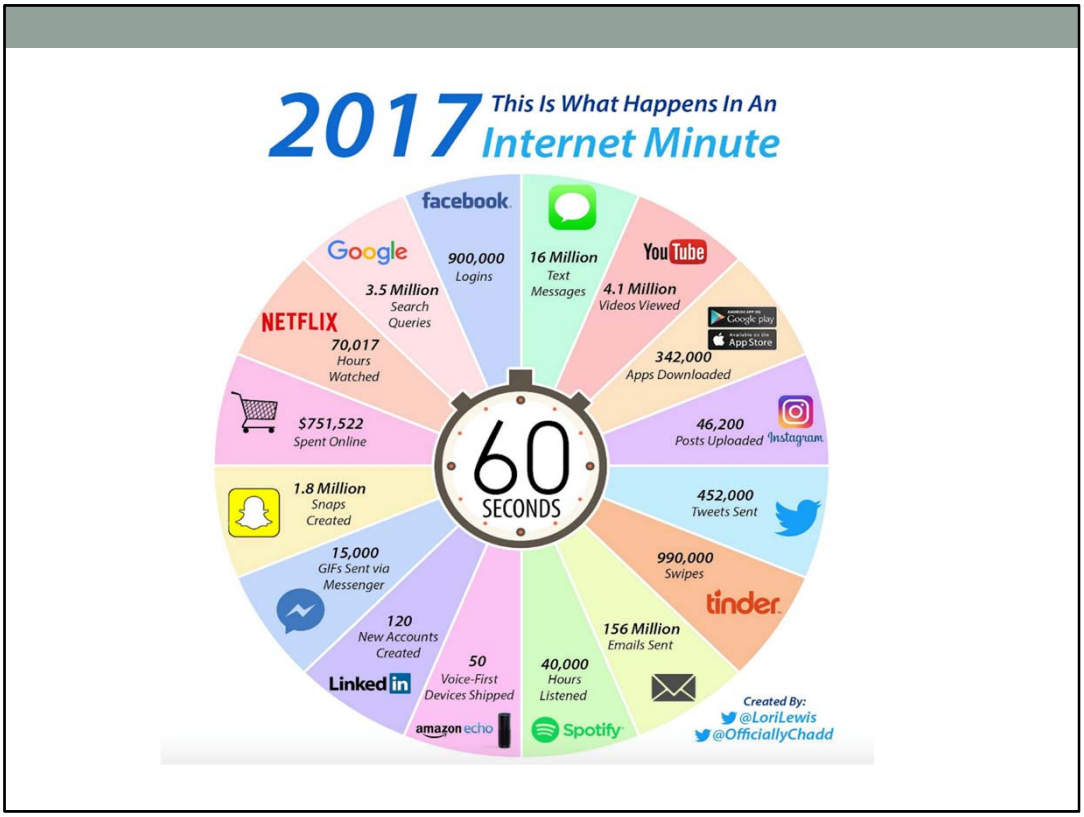
Bibliografía

***A Introduction to Information Retrieval**:*
Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze.
*Cambridge University Press, **2009**.*
Capítulos 19, 20 y 21

Contenidos

1. Introducción a la RI en la web
2. Publicidad en los buscadores
3. Detección de contenidos duplicados
4. Web Crawler
5. La web como un grafo dirigido
6. Uso del texto de los enlaces
7. Page-Rank
8. HITS

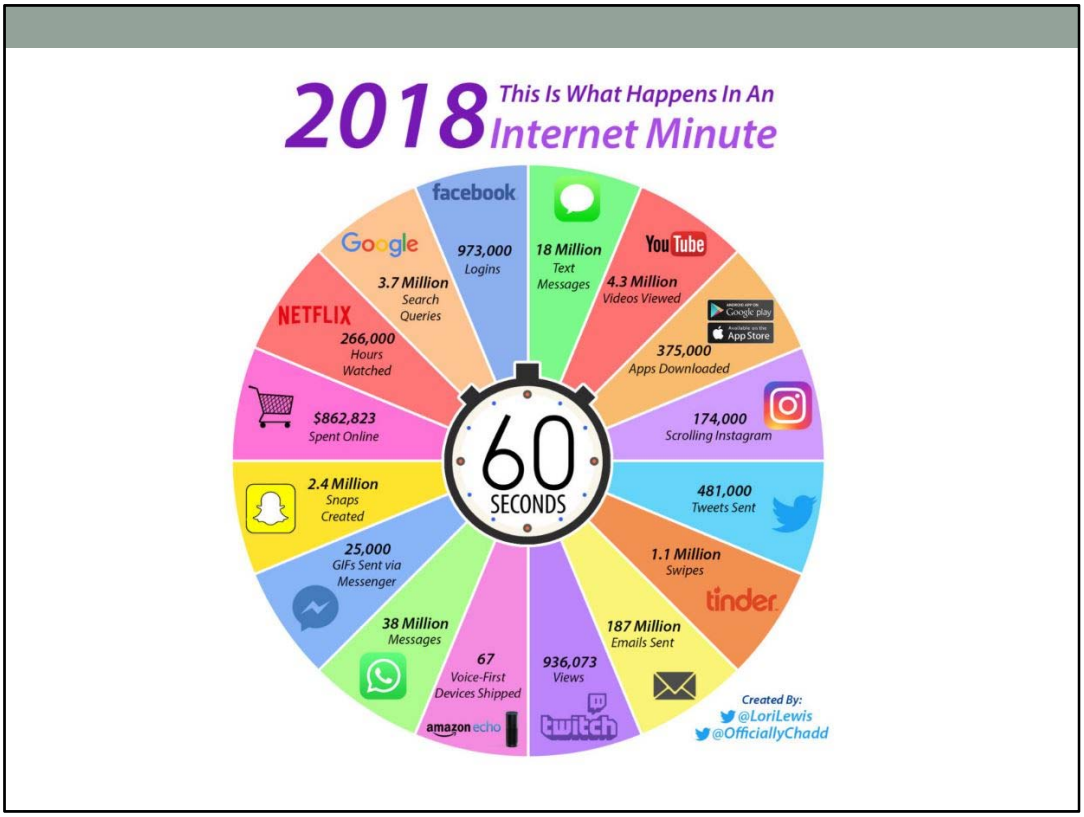
1. INTRODUCCIÓN A LA RI EN LA WEB



El tamaño de la web.

Lo que ocurre en un minuto en internet

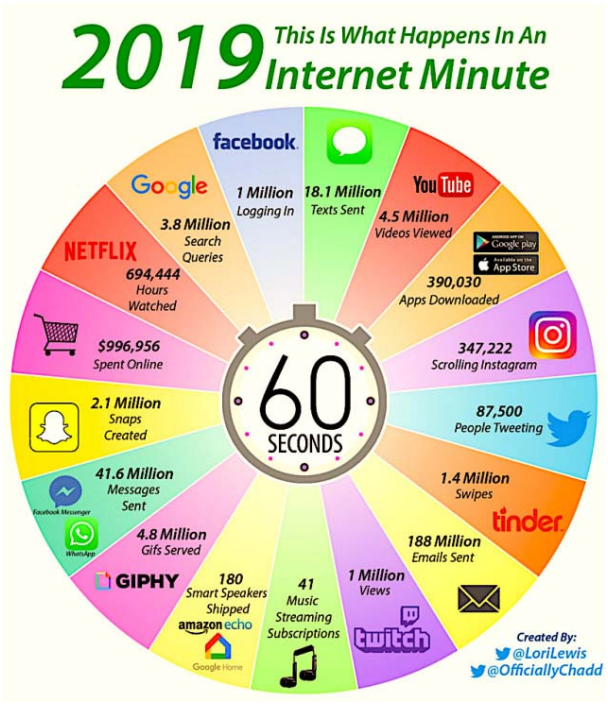
Más de 2 millones de búsquedas en google



El tamaño de la web.

Lo que ocurre en un minuto en internet.

Más de 3 millones de búsquedas en google.



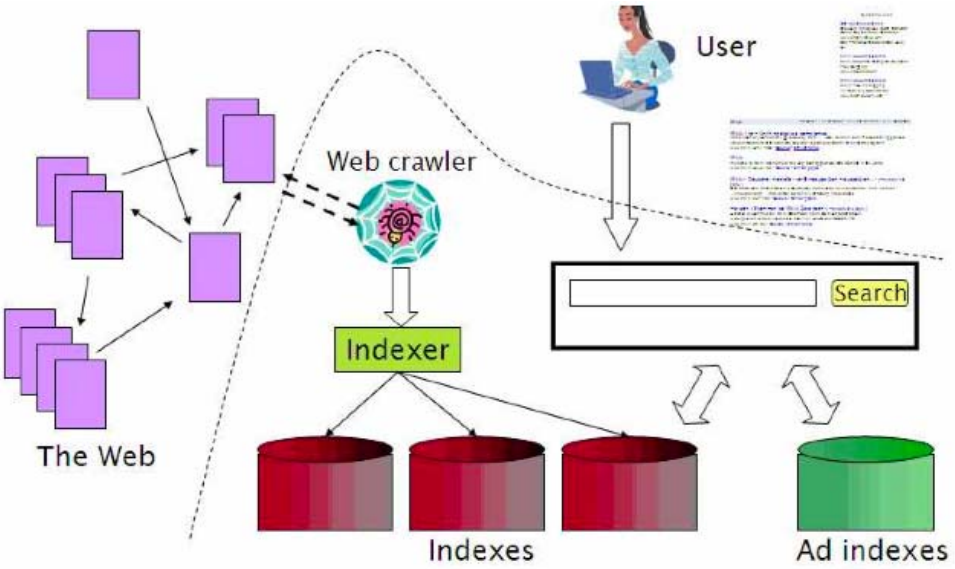
El tamaño de la web.

Lo que ocurre en un minuto en internet.

Casi 4 millones de búsquedas en Google.

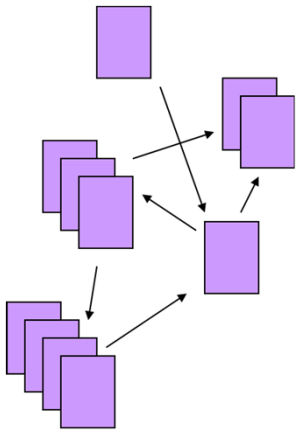
Se aprecia una fuerte subida de Netflix respecto del año anterior.

Esquema de la RI en la web



Esquema de la RI en la web.
Se muestra una imagen compuesta de un motor de búsqueda web que incluye el rastreador web (araña), así como los índices de anuncios y páginas web.
La parte de la figura debajo de la línea discontinua curva es interna al motor de búsqueda.
El rastreo web es el proceso mediante el cual recopilamos páginas de la Web, para indexarlas y alimentar el motor de búsqueda.
El término SERP (Search Engine Results Page) hace referencia a la página de resultados del buscador.

La colección de documentos web



- No hay ninguna coordinación en el diseño.
- Creación de contenido distribuido, todo el mundo puede crear contenido.
- Presencia de cientos de lenguas (ausencia de muchas otras)
- El contenido incluye verdades, mentiras, información obsoleta, contradicciones, …
- Información no estructurada (text, html, …), semi-estructurada (XML, annotated photos), estructurada (Databases)…
- Una escala mucho mayor que las colecciones anteriores…
- En continua expansión
- El contenido puede ser generado dinámicamente.

La Web no tiene precedentes en muchos sentidos: sin precedentes en escala, sin precedentes en la casi completa falta de coordinación en su creación, y sin precedentes en la diversidad de antecedentes y motivos de sus participantes. Cada una de estas características contribuye a hacer que la búsqueda en la web sea diferente, y en general mucho más difícil, que la búsqueda de documentos "tradicionales".
Los primeros buscadores se centraron en adaptar las técnicas de RI a un tamaño mucho más grande y en eso tuvieron éxito. Pero los resultados de las búsquedas no eran los mejores debido a la propia idiosincrasia de la web. Se necesitan métricas diferentes.

La web necesita buscadores

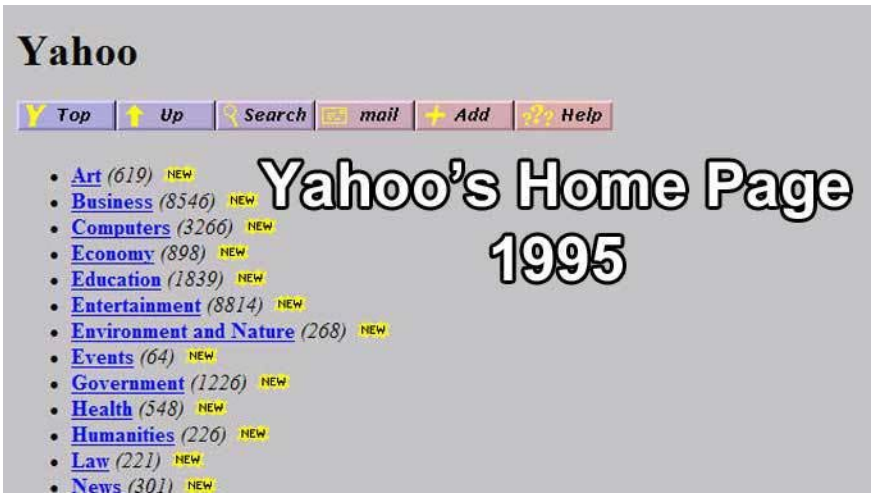
- Sin un buscador sería muy difícil encontrar “nuevas” páginas web.
- Si las webs no se encuentran no hay incentivo para crear contenido.
  - ¿Por qué publicar algo si nadie va a leerlo?
  - ¿Por qué publicar algo si no obtendremos ingresos (por publicidad) por ello?
- La web es gratis pero cuesta mucho dinero. Alguien tiene que pagar.
  - Servidores, infraestructura web, creación de contenido, …
- La publicidad en las búsquedas (y en las páginas web) está pagando gran parte de la web.

Primeros intentos de hacer visible la web a los usuarios

- Basados en el uso de taxonomías.
  - Agrupaban las webs en categorías.
  - **Yahoo! (dir.yahoo.com)**, **about.com**, **Open Directory Project**.
  - No pretendían clasificar toda la web. Sólo las “mejores” web de cada categoría.
- Basados en técnicas clásicas de recuperación de información.
  - **Altavista**, **Excite**, **Infoseek**.
  - Intenta indexar toda la web.

Los primeros intentos de hacer que la información de la web fuese "detectable" para otros usuarios se puede dividir en dos grandes categorías:
(1) motores de búsqueda basados en el uso de taxonomías agrupando las páginas web en categorías: como Yahoo!, About.com y el Open Directory Project
(2) motores de búsqueda basados en indizar el texto completo de las web: Altavista, Excite, Infoseek

Primeros intentos de hacer visible la web a los usuarios

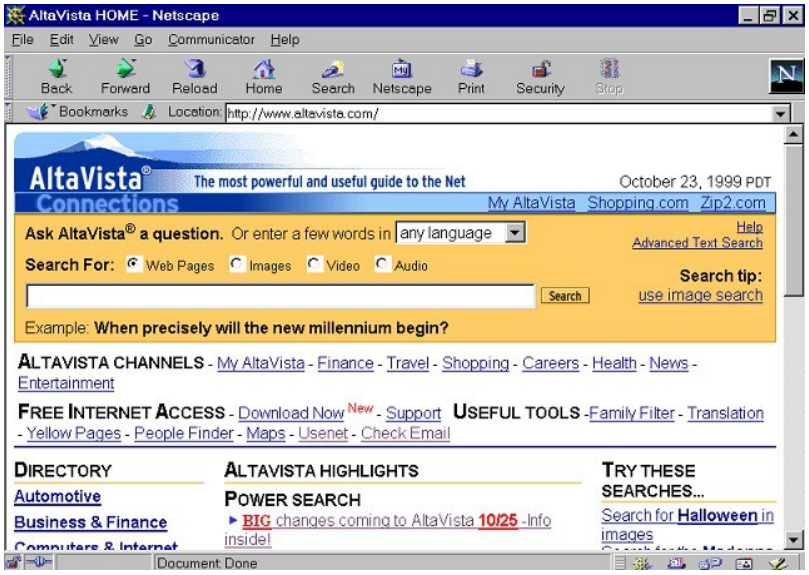


Primeros intentos de hacer visible la web a los usuarios

- Basados en el uso de taxonomías.
  - **Yahoo!**, **about.com**, **Open Directory Project**.
  - Agrupaban las webs en categorías.
  - No pretendían clasificar toda la web. Sólo las "mejores" web de cada categoría. **Yahoo!** tenía más de 10.000 categorías distintas.
  - Necesitaban un proceso manual de edición: "descubrimiento" de contenidos y elección de la categoría dentro de la taxonomía.
  - El usuario sólo "encontraba" la web si la buscaba en la categoría en la que el editor decidía clasificarla.

Los buscadores basados en el uso de taxonomías permitían al usuario navegar a través de un árbol jerárquico de etiquetas de categorías.
Aunque es un método cómodo e intuitivo para encontrar páginas web, tiene inconveniente. En primer lugar, la clasificación con precisión de las páginas web en los nodos de árbol de la taxonomía es, en su mayor parte, un proceso manual de la editorial, que es difícil de escalar al tamaño de la web (realmente solamente se incluía las mejores páginas web para cada categoría).
Sin embargo, sólo descubrir y clasificar con precisión estas "pocas" webs requiere un gran esfuerzo humano.
En segundo lugar, para que un usuario encuentre una web debe buscarla en el mismo subárbol de la taxonomía en el que el editor decidió clasificarla.
El tamaño de la taxonomía creció demasiado; el árbol de la taxonomía de Yahoo! superó bastante pronto los 1000 nodos distintos.

Primeros intentos de hacer visible la web a los usuarios



Primeros intentos de hacer visible la web a los usuarios

- Basados en técnicas clásicas de recuperación de información.
  - **Altavista**, **Excite**, **Infoseek**.
  - Intenta indexar toda la web.
  - Centraron los esfuerzos en solucionar problemas de escala: millones de documentos a indexar.
  - Los resultados de las búsquedas no eran buenos.
  - Necesidad de nuevos medidas de ranking de los resultados.

Los buscadores basados en técnicas de recuperación de información “clásica”, es decir, en la búsqueda de palabra clave usando índices invertidos y mecanismos de clasificación vistos en capítulos anteriores.
Los primeros buscadores se centraron en adaptar las técnicas de RI a un tamaño mucho más grande y en eso tuvieron éxito. Pero los resultados de las búsquedas no eran los mejores debido a la propia idiosincrasia de la web.

Se necesitan métricas diferentes.
En la RI clásica se mide lo cerca que está el documento de la consulta.
En la RI en la web se debe medir lo “confiable” que es la web que aloja el contenido.

2. LA PUBLICIDAD EN LOS BUSCADORES

Al principio de la historia de la búsqueda web, quedó claro que los motores de búsqueda web eran un medio importante para conectar a los anunciantes con posibles compradores.
Al principio de la historia de la Web, las compañías usaban anuncios gráficos de banner en páginas y sitios web populares. El objetivo principal de estos anuncios era la marca: transmitir al espectador un sentimiento positivo sobre la marca de la empresa que coloca el anuncio.

1ª generación de buscadores: resultados guiados por la publicidad

www.goto.com (1996)

Un ejemplo de la primera generación de buscadores web: GOTO.

1ª generación de buscadores: resultados guiados por la publicidad

- Buddy Blake hizo la puja más alta ($0,38) por esta búsqueda.
- Pagaba $0,38 cada vez que alguien hacía clic en el enlace.
- Páginas se clasificaban según lo que los anunciantes pagaban a **goto**.
- No hay separación entre los resultados y la publicidad
- No hay ranking de relevancia. Pero **goto** era honesto.

Para cada término de la consulta q GOTO aceptaba ofertas (pujas) de compañías que querían que su página web se mostrara en la consulta q.
En respuesta a la consulta q, GOTO devolvería las páginas de todos los anunciantes que ofertaron por q, ordenadas por sus ofertas.
Cuando el usuario hizo clic en uno de los resultados devueltos,
el anunciante correspondiente haría un pago a GOTO (en la implementación inicial, este pago equivalía a la oferta del anunciante por q).
Este estilo de motor de búsqueda se conoció de como “sponsored search” o “search advertising”.

2ª generación de buscadores: separación entre resultados y publicidad

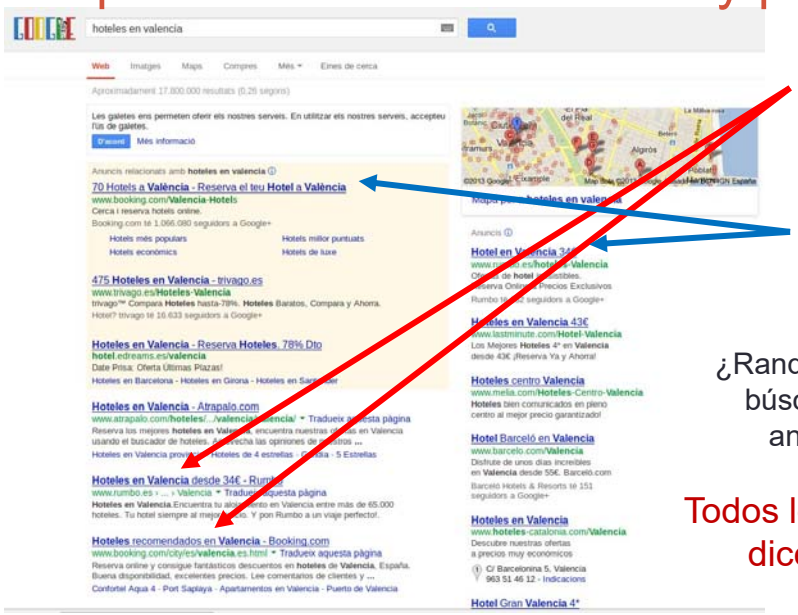
- Los buscadores separan en diferentes listas:
  - Resultados de las búsquedas, *algorithmic search results*
  - La publicidad de los anunciantes, *sponsored search results*

El ser anunciante de un buscador no proporciona mayor prioridad a la hora de aparecer más alto en los resultados del buscador.

Recuperar resultados de búsqueda patrocinados y clasificarlos en respuesta a una consulta ahora se ha vuelto considerablemente más sofisticado que el simple esquema GOTO.

Los motores de búsqueda actuales siguen en cierto modo este modelo: proporcionan un resultado de búsqueda puro (generalmente conocido como resultados de búsqueda algorítmica) como la respuesta principal a la búsqueda de un usuario, junto con los resultados de búsqueda patrocinados que, generalmente, se muestran por separado.

2ª generación de buscadores: separación entre resultados y publicidad



booking y **rumbo** aparece en los resultados de la búsqueda

booking y **rumbo** aparecen en la publicidad

¿Ranquean mejor los búscadores a los anunciantes?

Todos los buscadores dicen que **NO**

Un ejemplo del resultado de una búsqueda en Google de la consulta "hoteles en Valencia" (actualmente Google ha cambiado el aspecto de su SERP).
¿Ranquean mejor los búscadores a los anunciantes?
Todos los buscadores dicen que NO.

¿Influyen los anunciantes en el contenido editorial?
Un problema similar se plantea en periódicos y canales de TV y radio.
Los periódicos son reacios a publicar duras críticas de sus principales anunciantes.

¿Cómo se ordena la publicidad?

- Los anunciantes pujan por términos de búsqueda.
- Es un sistema abierto: Cualquier persona puede “comprar” de forma no exclusiva un término.
- Diferentes formas de pago:
  - Cost Per Click (CPC): El anunciante paga sólo cuando los usuarios hacen clic en el enlace.
  - Cost Per Mille (CPM): El anunciante paga una cantidad por cada 1000 impresiones de su anuncio en la página de resultados (SERP).
  - Cost Per Action (CPA): El anunciante paga en función de los usuarios que realmente finalizan una acción: comprar, registrarse, …

adwords.google.com
bingads.microsoft.com

Cada vez que alguien hace una búsqueda en Google, AdWords realiza una subasta para determinar los anuncios que se mostrarán en la página de resultados de la búsqueda (SERP) y su clasificación a la página.
Si desea que sus anuncios participen en esta subasta, debe decidir el tipo de acción del cliente por la que desea pagar.
Hay definidas diferentes formas de pago.

¿Cómo se ordena la publicidad?

- En cada búsqueda de los usuarios el buscador decide qué anuncios aparecerán y en qué orden.
- La elección no depende sólo de la puja del anunciante.
- Se tiene en cuenta la relevancia del anuncio.
- Factores que intervienen en la relevancia: la query, la zona horaria, la ubicación de usuario, la velocidad de carga de la página, …
- Una medida de la relevancia de un anuncio, el ratio de cliqueo Click-Through Ratio (CTR).

$$CTR = \frac{número\ de\ clics}{número\ de\ impresiones} \cdot 100$$

- El CTR varía por muchos factores (0.1% o 0.3% se puede considerar normal).

El CTR es el porcentaje de veces que el usuario hace clic en el anuncio de todas las veces que el anuncio se imprime (se muestra en pantalla).
Un 0,1 y 0,3 de CTR se considera normal.

¿Cómo se ordena la publicidad?

- En cada búsqueda de los usuarios el buscador decide qué anuncios aparecerán y en qué orden.
- Para los anunciantes, entender cómo los buscadores deciden este ranking, y qué pujas hacer sobre las diferentes keywords y a los diferentes sponsored search engines ha pasado a ser una profesión conocida como *Search Engine Marketing* (SEM).
- De forma paralela se define *Search Engine Optimization* (SEO) como la disciplina que se encarga del posicionamiento de un sitio web en los buscadores, con el objetivo de mejorar su visibilidad. En este caso hablamos de los resultados de la búsqueda algorítmica en los índices de las páginas web.

SEM es la disciplina que trabaja con el marketing en buscadores.
Es decir, se encarga de las campañas online en los buscadores y hacerlas visibles en los resultados patrocinados de dichos motores de búsqueda como, por ejemplo Adwords. Dentro de las estrategias SEM forman parte la gestión de palabras clave o la gestión de pujas.

SEO es la disciplina que trabaja con posicionamiento en la búsqueda algorítmica. Engloba el trabajo que se realiza sobre un sitio web y sus contenidos para que los motores de búsqueda lo encuentren y detecten si es suficientemente relevante para mostrárselo al usuario.

Publicidad en los buscadores

- Los buscadores reciben la gran mayoría de ingresos por publicidad.
- En el caso más general el anunciante sólo paga cada vez que el usuario hace clic a su enlace.
- El usuario sólo hace clic si está interesado en el anuncio.
- Los buscadores castigan anuncios fraudulentos e irrelevantes y presentan los más atractivos en las primeras posiciones.
- El usuario normalmente queda satisfechos al hacer clic en un anuncio.
- El anunciante encuentra a nuevos clientes de manera rentable.
- Todos contentos…

3. DETECCIÓN DE CONTENIDOS DUPLICADOS

Detección de duplicados

La web esta llena de contenidos duplicados.

Muchos más duplicados que en cualquier otra colección de documentos.

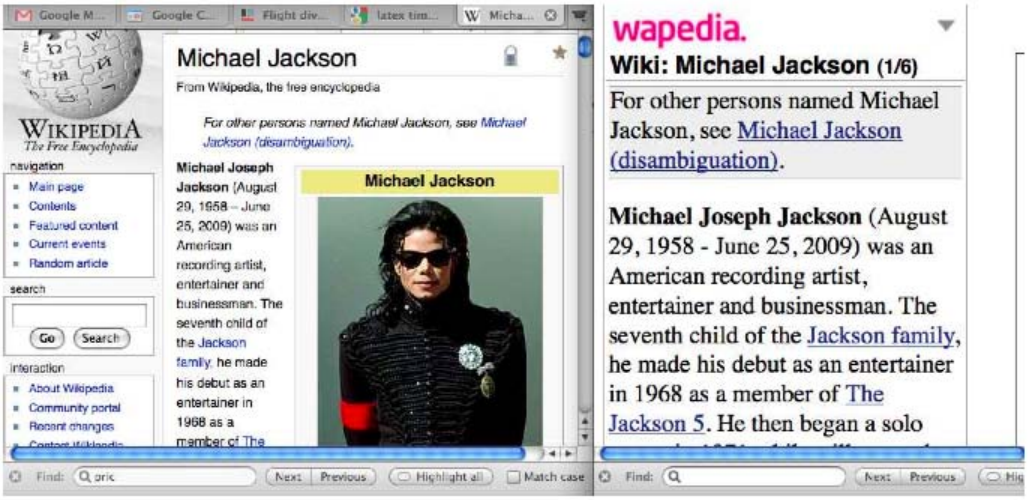
- Duplicados exactos.
  - Fáciles de detectar.
  - Hash o fingerprints
- Documentos casi iguales (near-duplicates).
  - Muy abundantes en la web.
  - Difíciles de detectar.

Es importante que documentos casi idénticos no aparezcan juntos en los resultados de una búsqueda.

Se deben detectar los documentos casi iguales.

La web está llena de contenido duplicado. Mucho más que cualquier otra colección de documentos.
Muchos de éstos son copias legítimas; por ejemplo, ciertos repositorios de información se duplican (mirrored) simplemente para proporcionar redundancia y confiabilidad de acceso.
Los motores de búsqueda intentan evitar indexar múltiples copias del mismo contenido, para mantener bajos los sobrecostes de procesamiento y almacenamiento.
Para el usuario, es molesto obtener un resultado de búsqueda con documentos idénticos cercanos.
Un documento de gran relevancia se convierte en irrelevante si aparece cerca de otro que es su duplicado.
Los duplicados exactos son fácilmente detectables, la aproximación más simple consiste en calcular para cada página web su “fingerprint”.
Una vez tokenizada, se aplica hash a sus elementos para representarla con un entero de, por ejemplo, 64 bits. Cuando los fingerprints de dos páginas resultan iguales, entonces se hace la comprobación de igualdad.
Este enfoque simplista no logra capturar un fenómeno crucial y generalizado en la web: la casi-duplicación.
También son muy abundantes los casi-duplicados en la web, y son bastante más difíciles de detectar.
Necesitamos eliminar los casi-duplicados.

Detección de contenidos casi iguales



¿Cómo detectar contenido casi idénticos?

Wapedia es una web que permite consultar los artículos de la wikipedia desde dispositivos móviles.

El contenido es casi idéntico al de la wikipedia pero puede haber diferencias, sería un ejemplo de documento casi-duplicado.

Detección de contenidos casi iguales

- Cuando hablamos de contenidos casi iguales nos referimos a similitud sintáctica.
- La similitud semántica, que dos webs digan lo mismo con palabras distintas, es mucho más difícil de detectar.

- Una primera aproximación a la detección de la similitud sintáctica podría ser el uso de una distancia de edición.
  - Podemos considerar que dos documentos son casi iguales si la similitud entre ellos es mayor que un cierto umbral θ (por ejemplo, 80%).

Vamos a estudiar una alternativa al uso de la distancia de edición para la detección de documentos casi-duplicados.

Similitud basada en shingles

- Un shingle es un n-grama del documento: una secuencia de n palabras que aparecen juntas en el documento.
- Dada una ***n*** los shingles de un documento es el conjunto de n-gramas de ese documento.

- Ejemplo: dado el texto "a rose is a rose is a rose" obtener el conjunto de shingles para n = 2, n = 3 y n= 4.
  - 2-gramas: { "a rose", "rose is" , "is a"}
  - 3-gramas: { "a rose is", "rose is a" , "is a rose" }
  - 4-gramas: {"a rose is a", "rose is a rose" , "is a rose is" }

Definimos un n-grama para un cierto valor de **n** como una secuencia de **n** palabras consecutivas en un texto.
Dado un documento **d** y fijado un valor para el parámetro **n**, se definie el conjunto de shingles **S** del documento como el conjunto de n-gramas que se encuentran en el documento, hablaremos de **S(d)**.
Para encontrar los n-gramas de un texto basta con definir una ventana de longitud **n** y desplazarla en el texto con un incremento de 1 posición cada vez. Se anotan los diferentes n-gramas que se van encontrando en el texto y esto constituye el conjunto de shingles del texto.

El conjunto de shingles de un documento puede usarse como una representación abreviada del documento. Nótese que no tiene en cuenta la posicion de los shingles en el texto ni el número de apariciones de los mismos.

Similitud basada en shingles

- Los shingles se pueden utilizar para medir la similitud sintáctica entre dos documentos.
- Si dos documentos tienen el mismo conjunto de shingles podemos decidir que son iguales, o casi.
- Dados dos documentos d1 y d2, una medida de similitud entre los dos documentos: coeficiente de Jaccard de sus conjuntos de shingles para un determinado valor de n.

Coeficiente de Jaccard

El coeficiente de Jaccard es una medida de solapamiento de conjuntos.

Dados dos conjuntos no vacíos A y B, se define el coeficiente de Jaccard entre ambos conjuntos Jaccard(A, B) (o simplemente J(A, B)) como el cociente entre la talla de la intersección de ambos conjuntos y la talla de su unión.

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- No es necesario que A y B tengan la misma talla.
- El coeficiente de Jaccard siempre es un número entre 0 y 1.

¿Cómo calcular el coeficiente de Jaccard?

J(A, A) = 1

J(A, B) = 0 sii la interseccion de A y B es vacía

Coeficiente de Jaccard

Ejercicio: dados los documentos:

d1: “Jack London traveled to Oakland”

d2: “Jack London traveled to the city of Oakland”

d3: “Jack traveled from Oakland to the city of London”

Calcular el coeficiente de Jaccard para n = 2 y n = 3.

n = 2

S(d1) = {“Jack London”, “London traveled”, traveled to”, “to Oakland”}

S(d2) = {“Jack London”, “London traveled”, “traveled to”, “to the”, “the city” “city of”, “of Oakland”}

S(d3) = {“Jack traveled”, “traveled from”, “from Oakland”, “Oakland to”, “to the”, “the city”, “city of”, “of London”}

J(S(d1), S(d2)) = 3/8 = 0,375, J(S(d1), S(d3)) = 0, J(S(d2), S(d3)) = 3/12 = 0,25

Dados los tres documentos d1, d2 y d3, vamos a calcular sus conjuntos de shingles para n=2 y n=3.

Utilizaremos el coeficiente de Jaccard de sus conjuntos de shingles para medir la similitud entre los documentos.

Coeficiente de Jaccard

Ejercicio: dados los documentos:

d1: “Jack London traveled to Oakland”

d2: “Jack London traveled to the city of Oakland”

d3: “Jack traveled from Oakland to the city of London”

Calcular el coeficiente de Jaccard para n = 2 y n = 3.

n = 3

S(d1) = {“Jack London traveled”, “London traveled to”, traveled to Oakland”}

S(d2) = {“Jack London traveled”, “London traveled to”, “traveled to the”, “to the city”, “the city of”, “city of Oakland”}

S(d3) = {“Jack traveled from”, “traveled from Oakland”, “from Oakland to”, “Oakland to the”, “to the city”, “the city of”, “city of London”}

J(S(d1),S(d2)) = 2/7 = 0,286, J(S(d1),S(d3)) = 0, J(S(d2),S(d3)) = 2/11 = 0,182

¿Cómo calcular el coeficiente de Jaccard de forma eficiente?

Para n=3.

Calculo del coeficiente de Jaccard

Vamos a utilizar un hashing (por ejemplo de 64bits).

- A cada shingle le asociamos un valor de hash en un espacio de 64 bits.
- $H(d_j)$ es el conjunto de valores de hash derivado del conjunto de shingles $S(d_j)$.
- Sea π una permutación aleatoria de los enteros de 64 bits en los enteros de 64 bits.

Utilizamos una especie de hashing.

En primer lugar, asignamos a cada shingle un valor hash sobre un gran espacio, por ejemplo 64 bits.

Para **j = 1, 2**, sea **H(d_j)** el correspondiente conjunto de valores de hash de 64-bit derivado de **S(d_j)**.

Ahora hacemos el siguiente “truco” para detectar pares de documentos cuyos conjuntos **H** tienen un alto coeficiente de Jaccard.

Sea **π** una permutación aleatoria de los enteros de 64 bits a los enteros de 64 bits.

Calculo del coeficiente de Jaccard

Denotamos con $\Pi(d_j)$ el conjunto de valores hash permutados de $H(d_j)$; cada $h \in H(d_j)$ tendrá un valor $\pi(h) \in \Pi(d_j)$ asociado.

Sea x_j^{π} el menor entero de $\Pi(d_j)$.

Teorema:

$$J\big(S(d_1), S(d_2)\big) = P(x_1^{\pi} = x_2^{\pi})$$

Denotamos con **$\Pi(d_j)$** el conjunto de valores hash permutados en **$H(d_j)$**.

Para cada **$h \in H(d_j)$**, habrá un valor **$\pi(h) \in \Pi(d_j)$** asociado.

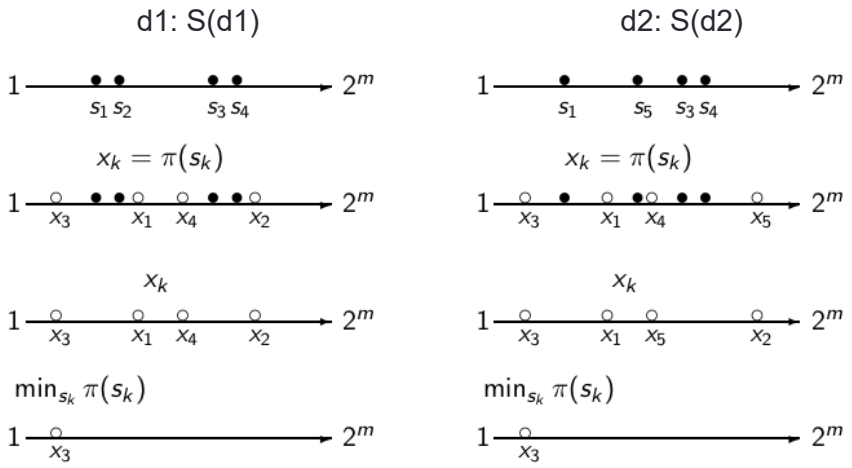
Sea **x^{π}_j** el menor entero de **$\Pi(d_j)$**.

Hay un teorema que asegura que el coeficiente de Jaccard entre los conjuntos **S(d1)** y **S(d2)** se puede calcular como la probabilidad de que **x^{π}_1** y **x^{π}_2** sean iguales.

Realmente el teorema es un poco más “complicado”.

Dice que el coeficiente de Jaccard es igual a la probabilidad de que en “cualquier permutación aleatoria” los dos mínimos sean iguales.

Calculo del coeficiente de Jaccard



Para la permutación π los documentos d1 y d2 son casi idénticos

Demostración del teorema

S(d1)	S(d2)
0	1
1	0
1	1
0	0
1	1
0	1

Sea la siguiente tabla una representación de los shingles de dos documentos:

- cada columna es un documento
- cada fila un shingle.
- cada celda tiene valor 1 o 0 en función de que el shingle pertenezca o no al conjunto de shingles del documento.

Vamos a describir los conjuntos de shingles con un vector binario, de forma que la primera fila es para el shingle con el menor valor de hash, la segunda para el shingle con el siguiente menor valor de hash, ... Para cada conjunto S(d1) y S(d2), en la fila correspondiente al shingle i-ésimo en la ordenación que en base al hash hemos realizado sobre los shingles, un valor 1 indica que ese shingle pertene al conjunto que estamos representando y 0 en caso contrario.

Demostración del teorema

S(d1)	S(d2)
0	1
1	0
1	1
0	0
1	1
0	1

Sea:

- C00: el número de filas donde en ambas columnas hay un 0.
- C01: el número de filas donde en la primera columna hay un 0 y en la segunda un 1.
- C10: el número de filas donde en la primera columna hay un 1 y en la segunda un 0.
- C11: el número de filas donde en las dos columnas hay un 1.

Introducimos la definición de contadores que se muestra en la diapositiva.

Para el ejemplo de dos documentos los valores son:

C00 = 1

C01 = 2

C10 = 1

C11 = 2

Demostración del teorema

S(d1)	S(d2)
0	1
1	0
1	1
0	0
1	1
0	1

¿Cuál es el coeficiente de Jaccard de los dos documentos?:

$$J(S(d1), S(d2)) = \frac{|S(d1) \cap S(d2)|}{|S(d2) \cup S(d2)|} = \frac{C11}{C01 + C10 + C11}$$

¿Cuál es la probabilidad de que en cualquier permutación aleatoria de filas el primer 1 aparezca simultáneamente en ambas columnas?

$$P(x_1^\pi = x_2^\pi) = \frac{C11}{C01 + C10 + C11}$$

$$\boxed{J\big(S(d_1), S(d_2)\big) = P(x_1^\pi = x_2^\pi)}$$

Cálculo de la probabilidad:

Una vez permutadas las filas nos ponemos en la primera fila:

- Vamos pasando de los 00 -> C00 no se tiene en cuenta
- Una vez encontrado el primer 1 en la fila puede ser

11 -> caso favorable, 01 -> caso desfavorable, 10 -> caso desfavorable

- La probabilidad es casos favorables / casos posibles

Calculo del coeficiente de Jaccard

- Generamos 200 permutaciones aleatorias, pueden ser 200 funciones de hash distintas.
- Para cada permutación π y cada documento d_i calculamos x_i^{π}. Llamamos ψ_i al conjunto de 200 valores de x_i^{π} para el documento d_i.

Aproximamos el coeficiente de Jaccard como:

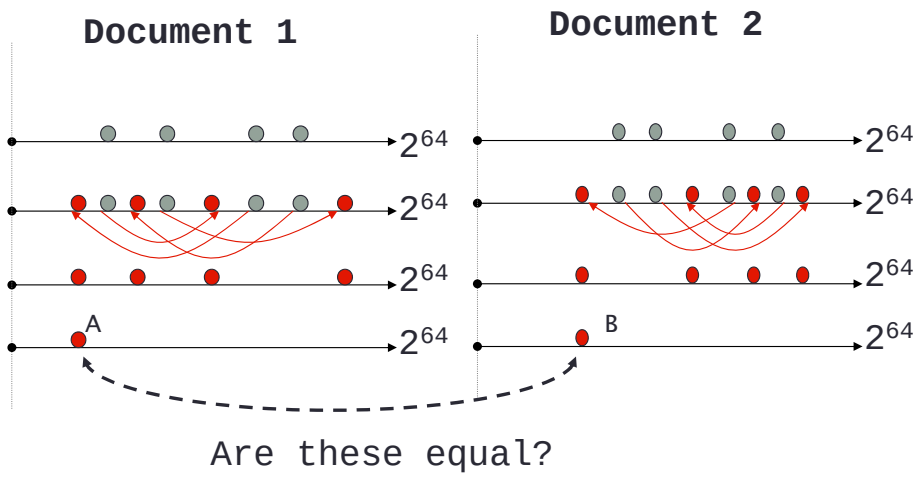
$$J(S(d_i), S(d_j)) \approx \frac{|\psi_i \cap \psi_j|}{200}$$

Si el valor obtenido supera un umbral determinado podemos afirmar que los documentos d_i y d_j son similares.

Para simular la condición del teorema de "cualquier permutación aleatoria", se generan 200 permutaciones, que pueden simularse con el uso de 200 funciones hash distintas.
Para un documento dado y para cada una de las 200 permutaciones aleatorias se calcula el valor mínimo.
Dados dos documentos y sus respectivos conjuntos de valores menores calculados, se aproxima el coeficiente de Jaccard entre los conjuntos de singles como se indica en la fórmula de la diapositiva.
Si el valor obtenido supera un umbral determinado podemos afirmar que los documentos $\boldsymbol{d_i}$ y $\boldsymbol{d_j}$ son similares.

El número "mágico" de 200 permutaciones aparece reflejado en la literatura.
Esto no calcula realmente el coeficiente de jaccard pero se le parece bastante.

Calculo del coeficiente de Jaccard



Testear para 200 permutationes random: $\pi_1, \pi_2, \ldots \pi_{200}$

El número “mágico“ de 200 permutaciones aparece reflejado en la literatura.

Esto no calcula realmente el coeficiente de jaccard pero se le parece bastante.

Calculo del coeficiente de Jaccard

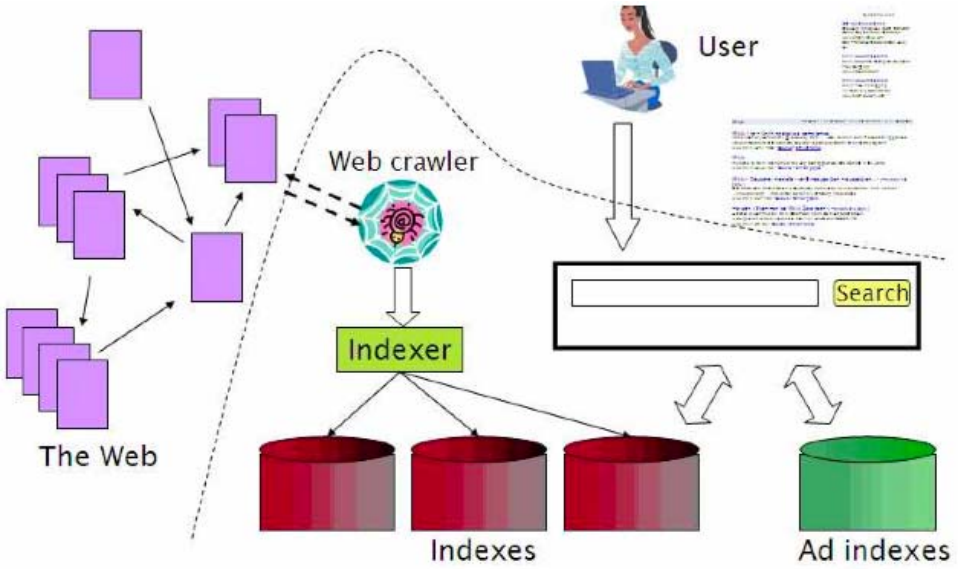
Una última optimización heurística para reducir el número de pares de documentos a los que calcular el coeficiente de similitud:

- Ordenar los elementos de ψ_i y calcular su conjunto de shingles (super-shingles).
- Sólo compararemos (utilizando el coeficiente de Jaccard) aquellos documentos que tengan algún super-shingle en común.

Incluso con todas las optimizaciones propuestas, se deben aplicar más heurísticos para el calculo de similitud de documentos en el mundo real.

4. WEB CRAWLER

Esquema de la RI en la web



El rastreo web es el proceso mediante el cual recopilamos páginas de la Web, para indexarlas y alimentar el motor de búsqueda.

Denominaciones:

- Web crawler,
- Web spider,
- Web ant,
- Robot.

La más utilizada en español es araña web y rastreador.

Web crawler

Los sistemas de recuperación de información deben acceder al contenido de los documentos para indexarlos.

Sistema de recuperación de información clásico:

- Fácil y rápido.
- Indexar una colección de documentos en un directorio de un disco local.
- Hacer un recorrido recursivo por el sistema de ficheros.

En un sistema de RI clásico sabemos donde están los documentos, suele ser una colección fija y determinada a priori.

Web crawler

Los sistemas de recuperación de información deben acceder al contenido de los documentos para indexarlos.

Sistema de recuperación de información para la web:

- Más complicado y mayor coste temporal (latencia).
- Indexar una colección de documentos repartidos por múltiples servidores.
- Analizar los enlaces de los documentos para recuperar nuevos documentos.

En un sistema de RI para la web el análisis de los documentos va "descubriendo" nuevos documentos.

- Se tarda mucho más tiempo en acceder a los documentos.
- La infraestructura que soporta los documentos no es "nuestra".

No debemos "molestar"

¿Pero es realmente un desafío para los diseñadores de sistemas?

Sí es una parte crucial para que el rendimiento (eficiencia y eficacia) del buscador sea bueno.

¿Qué debe hacer un web crawler?

Inicializar la cola con URLs de páginas conocidas

Repetir:

- Coger una URL de cola
- Recuperar y analizar la página
- Extraer las URLs de la página
- Añadir las URLs a la cola

Asunción: la web es un grafo fuertemente conexo.

El funcionamiento básico de cualquier rastreador de hipertexto (ya sea para la Web, una intranet u otra colección de documentos de hipertexto) es como sigue.
El rastreador comienza con una o más URL que constituyen un conjunto de semillas que se han almacenado en una cola de URLs.
Elige una URL de esta cola, luego busca la página web en esa URL.
La página recuperada se analiza para extraer tanto el texto como los enlaces de la página.
El texto extraído se proporciona al indexador de texto.
Los enlaces extraídos (URL) se agregan a la cola de URLs, que en todo momento consiste en URLs cuyas páginas correspondientes aún no han sido obtenidas por el rastreador.
Inicialmente, la cola de URLs contiene el conjunto de semillas; a medida que se obtienen las páginas, las URLs correspondientes se eliminan de la cola de URLs.
Todo el proceso puede verse como un recorrido del grafo web.

Con más detalle …

```
urlqueue := (urls iniciales bien seleccionadas)
while urlqueue is not empty:
    myurl := urlqueue.getlastanddelete()
    mypage := myurl.fetch()
    fetchedurls.add(myurl)
    newurls := mypage.extracturls()
        for myurl in newurls:
            if myurl not in fetchedurls
              and not in urlqueue:
                urlqueue.add(myurl)
    addtoinvertedindex(mypage)
```

¿Qué le falta al web crawler básico?

- Escalabilidad: Es necesario distribuir el proceso.
- Selección: No se puede indexar toda la web, hay que seleccionar lo que se va a indexar.
- Control de Duplicados. La detección de duplicados debe formar parte del proceso de crawling.
- Detección de la páginas de spam y "spider traps". Se deben detectar este tipo de páginas para no indexarlas y "liberar" el web crawler.

El crawler anterior no funcionaria en un sistema real: ¿qué es lo mínimo que le faltaría?

- Escalabilidad: tenemos que distribuir el proceso de crawling en diferentes máquinas

y permitir que se pueda seguir escalando de tal forma que reducir el tiempo (más potencia) sea añadir más máquinas.

- No podemos indexar todo: es necesario que seleccione. ¿Cómo determinamos qué se indexa y qué no?
- Control de Duplicados: necesitamos integrar el control de duplicados dentro del proceso de crawling.
- Detección de spam y "spider traps". Las spider traps son páginas web que suponen literalmente una trampa para el crawler. El crawler que queda iterando en ellas indefinidamente.

Pueden ser: maliciosas (programadas con el objetivo de atrapar a la araña), para intentar aumentar el número de enlaces a la página o sin intenciones maliciosas por el propio carácter dinámico de las páginas.

¿Qué le falta al web crawler básico?

- Cortesía: No se deben sobrecargar los servidores que alojan las páginas. Las peticiones a un mismo servidor deben espaciarse.
- Refresco: El contenido de las páginas se va actualizando. Necesitamos repetir el proceso de crawling periódicamente.
  - Sólo podemos hacer refrescos frecuentes de algunas páginas.
  - Problema de selección y priorización.

Ejemplo: Para obtener 25,000,000,000 páginas en un mes se deben procesar casi 10,000 páginas por segundo.

- Cortesía: no podemos sobrecargar la infraestructura que soporta el
- contenido que queremos indexar. Necesitamos ser “nice" y espaciar
- todas las peticiones de un sitio a largo plazo (horas, días).

Hay una directriz en robots.txt para indicar al crawler cuanto tiempo debe esperar entre cada acceso (lo veremos enseguida)

- Refresco: necesitamos volver a hacer crawling periódicamente.
- Debido al tamaño de la web, solamente podemos hacerlo de un
- pequeño subconjunto. Otra vez, problema de selección y priorización.

Debemos priorizar algunos refrescos. Algunas páginas se revisitan con más frecuencia que otras.

- Ejemplo: Para obtener 25,000,000,000 de páginas en un mes se deben procesar casi 10,000 páginas por segundo. En realidad: muchas más ya que muchas de las páginas que descarguemos serán spam o contenidos duplicados, etc..

robots.txt

- Definido en 1994.
- Un fichero de texto que se pone en la raíz del sitio web.
- Da indicaciones al crawler (también llamado robot) sobre qué parte de la propia web no debe ser indexada.
- Es decisión del crawler respetar las directrices de robots.txt.
- Los web crawler usados por los buscadores web respetan las directrices del robots.txt.

Muchos hosts en la Web colocan ciertas partes de sus sitios web fuera del alcance del rastreo, bajo un estándar conocido como el Protocolo de Exclusión de Robots. Esto se realiza colocando un archivo con el nombre robots.txt en la raíz de la jerarquía de URL en el sitio web.

Cualquiera puede hacer un crawler por tanto no esta garantizado que todos los crawlers respeten lo que pone en robots.txt

Un ejemplo real: www.upv.es/robots.txt

```
User-agent: *
Disallow: /upvrenew
Disallow: /wniujom
Disallow: /pls/
Disallow: /pls/soalu
Disallow: /pls/oalu
Disallow: /pls/sobib
Disallow: /pls/soarc
Disallow: /obibproxy
Disallow: /oaluproxy
Allow: /ical/*
Allow: /pls/ical/sic_ical_crypt.getCal*
Allow: /pls/obib/sic_bibpublicador.listas*
Allow: /pls/oreg/rtv_web.*
Allow: /pls/oalu/sic_per.info_persona*
```

Ejemplo de robots.txt (www.upv.es)

Crawl-delay: 10 -> espera 10 segundos entre peticiones al mismo servidor

content="noindex"

- content="noindex“ es un atributo que se añade a la etiqueta <meta> en la cabecera de un documento HTML.
- Indica a los web crawlers que no se indexe la página.

```
<html>
<head>
 <meta name="robots" content="noindex">
 <title>Esta página no se indexa</title>
</head>
```

Con este atributo se consigue que los robots no analicen la página.
Sirve para no indexar una página en un directorio donde si se quiere que se indexen las otras páginas y por tanto no se puede utilizar robots.txt.

Content puede ser: "none", "all", "index", "noindex", "nofollow", and "follow"

rel="nofollow"

- Definido en 2005 por Google y Blogger.
- rel="nofollow" es un atributo que se añade a la etiqueta de enlace <a> de HTML.
- El crawler sí sigue estos enlaces pero …
- Los enlaces con este atributo no son tenidos en cuenta por los buscadores al calcular los resultados de una consulta (en el cálculo del pagerank por ejemplo).
- Evitar el uso de spammers robots en los comentarios.
- Los spammers robots se siguen utilizando.

```
<a href="http://spam.com" rel="nofollow">texto</a>
```

Para los spammers era muy interesante añadir enlaces a sus páginas desde páginas con valores de ranking muy alto. Por ejemplo poner enlaces en blogs.
Existen spam robots que se dedican a añadir entradas en blogs (de forma automática) que contienen enlaces a sus páginas.
Al aumentar el número de enlaces, hace que la pagina con spam suba en el ranking del buscador.
Esta etiqueta se añade al tag <a> y hace que los crawlers no tengan en cuenta el enlace al calcular el pagerank, se desincentiva a los spammers.
PERO SÍ se analiza la página y el enlace. No es igual que noindex.
El spam en los comentarios no ha desaparecido porque los spammers siguen consiguiendo tráfico para sus webs.

Contenido ya visto

Para cada página descargada:

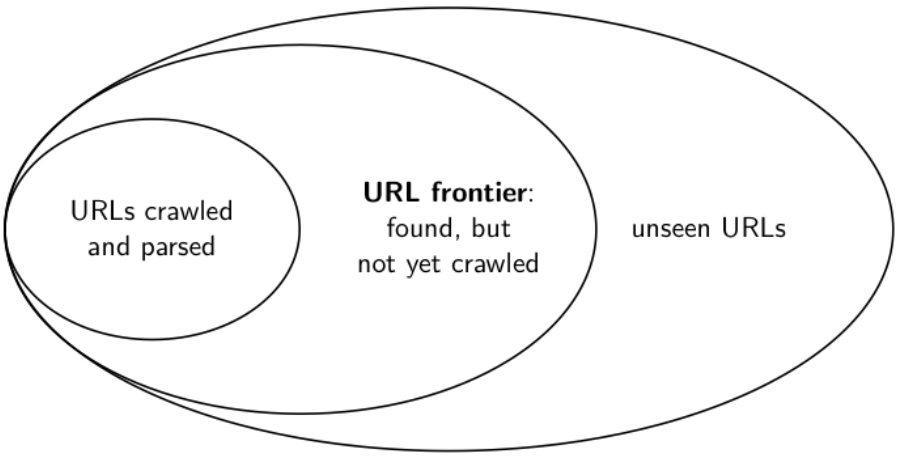
- Comprobar si el contenido es idéntico al de algún documento ya indexado (fingerprint).
- Comprobar si el contenido es casi igual al de algún documento ya indexado (shingles y coeficiente de Jaccard).
- Las páginas con contenido ya indexado deben descartarse.

Para cada página descargada debe realizarse una serie de verificaciones.

Normalización de URL

- Algunas URLs extraídas de un documento son las direcciones URL relativas.
- El enlace a info.html dentro de www.pagina.com es equivalente a un enlace a www.pagina.com/info.html.
- Todas las URLs relativas deben convertirse en absolutas durante el análisis.

URL frontier



Intuitivamente

La URL frontier es el conjunto de urls a páginas ya “descubiertas“ pero que todavía no han sido descargadas y analizadas.
Unseen URLs páginas que se supone (seguro) que existen pero de las que no sabemos nada, no han sido descubiertas todavía.

5. LA WEB COMO UN GRAFO DIRIGIDO

ES1

La web como un grafo dirigido

Las páginas web utilizan la etiqueta **<a>** (anchor) de HTML para enlazar a otras páginas mediante hipervínculos.

<**a** href="http://www.dirección_del_enlace.com">

texto del hipervínculo</**a**>

Podemos ver las páginas web estáticas junto a los hipervínculos entre ellas como un grafo dirigido en el que cada página web es un nodo y cada hipervínculo un arco dirigido.

Como ya hemos comentado, los primeros buscadores se centraron en adaptar las técnicas de RI a un tamaño mucho más grande y en eso tuvieron éxito. Pero los resultados de las búsquedas no eran los mejores debido a la propia idiosincrasia de la web.

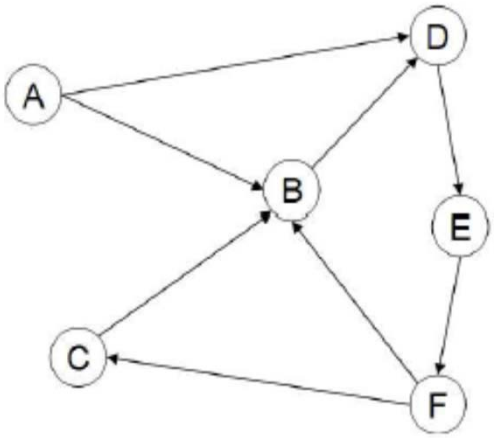
En un primer paso para la definición de nuevos métodos para medir la relevancia de una página web respecto de una consulta se estudia la web como un grafo dirigido.

Podemos ver las páginas web estáticas junto a los hipervínculos entre ellas como un grafo dirigido en el que cada página web es un nodo y cada hipervínculo un arco dirigido.

Diapositiva 59

ES1 Encarna Segarra; 13/03/2020

La web como un grafo dirigido



- Los nodos (**A, B, C, D, E, F**) representan páginas web.
- Los arcos representan hipervínculos entre las páginas.
- Ejemplo, **B** tiene:
  - grado de salida 1
  - grado de entrada 3

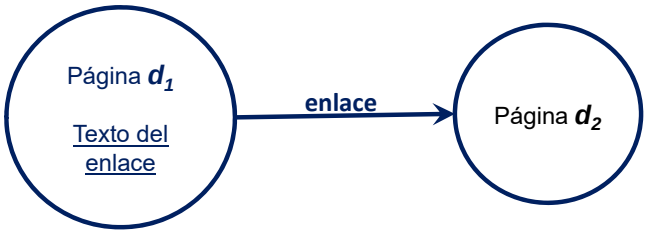
¿Este grafo es fuertemente conexo?

¿Es la web un grafo fuertemente conexo?

Veamos un pequeño ejemplo de grafo web.
En este ejemplo hay seis páginas etiquetadas de la A a la F.
La página B tiene un grado de salida (arcos de salida) igual a 1
y un grado de entrada (arcos de entrada) igual a 3.
En este ejemplo el grafo web NO ES fuertemente conexo:
no hay un camino desde ninguna web a A.

6. USO DEL TEXTO DEL ENLACE

La web es un grafo dirigido



Primera hipótesis: Un enlace es una señal de calidad.

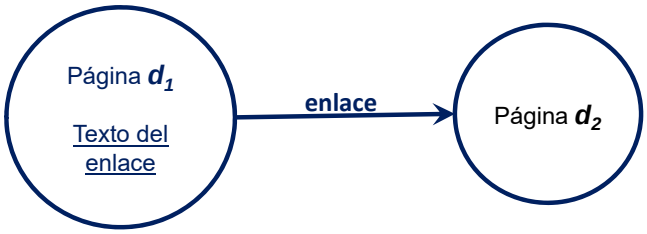
- El enlace de d_1 hacia d_2 indica que el autor de d_1 considera que d_2 es relevante y de alta calidad.

¿Es esto cierto habitualmente?

Siguiendo con el estudio de la web como un grafo dirigido, en este apartado plantemos la utilidad que el texto que aparece en los enlaces puede tener para la medida de la relevancia de la web enlazada.

En una primera hipótesis podemos interpretar un enlace de una página web $\mathbf{d_1}$ a una $\mathbf{d_2}$ como un indicador de relevancia: el autor de $\mathbf{d_1}$ considera que $\mathbf{d_2}$ es relevante y de alta calidad. Esta hipótesis es cierta en algunos casos, pero no siempre.

La web es un grafo dirigido



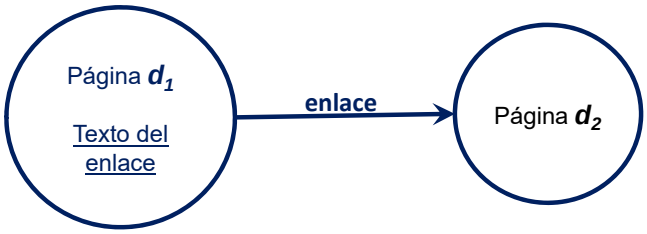
Segunda hipótesis: El texto del enlace de d_1 a d_2 es un buen descriptor del contenido de d_2.

- Aquí ampliamos el texto del enlace a texto "alrededor" del enlace.

¿Es esto cierto habitualmente?

La web está llena de ejemplos en los que el contenido de una página no representa una buena descripción de la misma.
Por lo tanto, a menudo hay una brecha entre los términos en una página web y cómo los usuarios web describirían esa página web.
Sin embargo, la descripción que otras páginas hacen de las páginas a las que enlazan sí que puede ser de utilidad.
Es por ello que una segunda hipótesis establece que el texto del enlace de $\mathbf{d_1}$ a $\mathbf{d_2}$ es un buen descriptor del contenido de $\mathbf{d_2}$.

La web es un grafo dirigido



Ejemplo:

¿Quieres cambiar de ordenador?. Los mejores ordenadores al mejor precio aquí

- Texto del enlace: Los mejores ordenadores al mejor precio aquí

Este concepto de descripción se extiende también al texto que aparece alrededor del enlace, ya que si consideramos estrictamente el texto del enlace (el que hay entre <a> y), en internet el más frecuente es “aquí”.

Uso del texto del enlace

- Un enlace representa una señal de calidad (primera hipótesis).
- El texto de los enlaces a una página web suele contener términos que describen adecuadamente la página. (segunda hipótesis).
- Muchas veces los términos de los enlaces no aparecen en la página.
- Pero muchas veces la describen mejor que el contenido de la propia página.

El texto de los enlaces describe de algún modo la visión que los usuarios de una página tienen de ella. Expresan en las palabras del usuario el contenido de la página.

Uso del texto del enlace

Ejemplo:

- Durante mucho tiempo la página oficial de IBM (www.ibm.com) no contenía la palabra IBM.
- La página oficial de IBM no contiene la palabra computer.
- Muchos de los enlaces a www.ibm.com contienen las palabras IBM o computer.

Un ejemplo de esto podría ser la página oficial de IBM.

Uso del texto del enlace

- Es una buena idea utilizar los términos del texto de los enlaces a una página para indexar esa página.
- Los términos de los enlaces son, muchas veces, tenidos más en cuenta por los buscadores que los términos de la página.
- Podemos encontrar una página buscando por términos que no aparecen en la página. Términos "no controlables" por el creador de la página.

Al indexar la página por el texto de los enlaces a ella ampliamos los términos que la hacen "accesible", pero perdemos el control sobre esos términos.
Normalmente es una buena idea aunque veremos casos en los que no lo es tanto (google bombs).
Utilizado de forma fraudulenta puede crear problemas.

Google bombs

- Intento de manipular los resultados de un buscador aprovechando que el buscador tiene muy en cuenta el texto de los enlaces a páginas web.
- Muchos enlaces a una misma página web desde múltiples sitios con un mismo texto.
- Al buscar por ese texto se consigue que se obtenga como resultado la página.
- En 2007 Google hizo cambios para minimizar el efecto.

Un Google bomb es una búsqueda con "malos" resultados debido a la manipulación maliciosa de los textos de los enlaces.

Una Google bomb es un esfuerzo cooperativo para manipular el motor de búsqueda al devolver un sitio web en particular como el primer resultado cuando se busca una palabra o frase específica.

El texto del enlace que alimenta la bomba puede dejarse en muchos lugares alrededor de la web: artículos o comentarios en blogs …

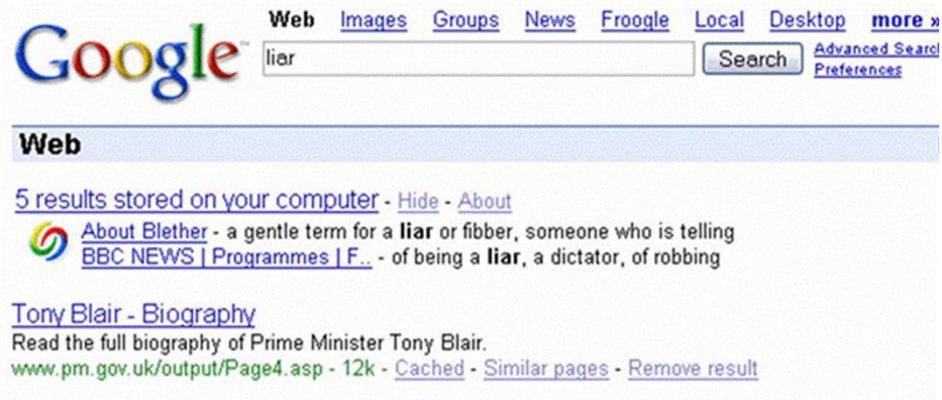
Google presentó una nueva función de ponderación en enero de 2007 para minimizar el efecto.

Google bombs

Algunas de las google bombs más sonadas tuvieron relación con la invasión americana de Iraq

Failure -> whitehouse

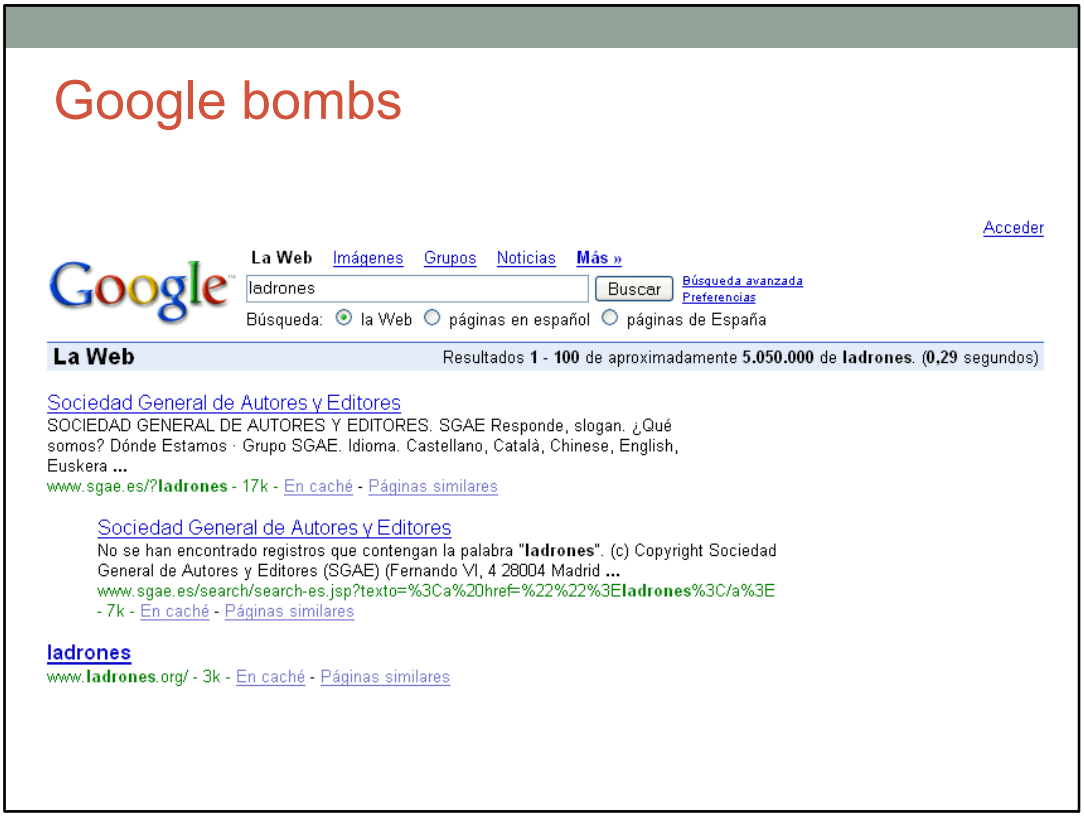
Google bombs



Algunas de las google bombs más sonadas tuvieron relación con la invasión americana de Iraq

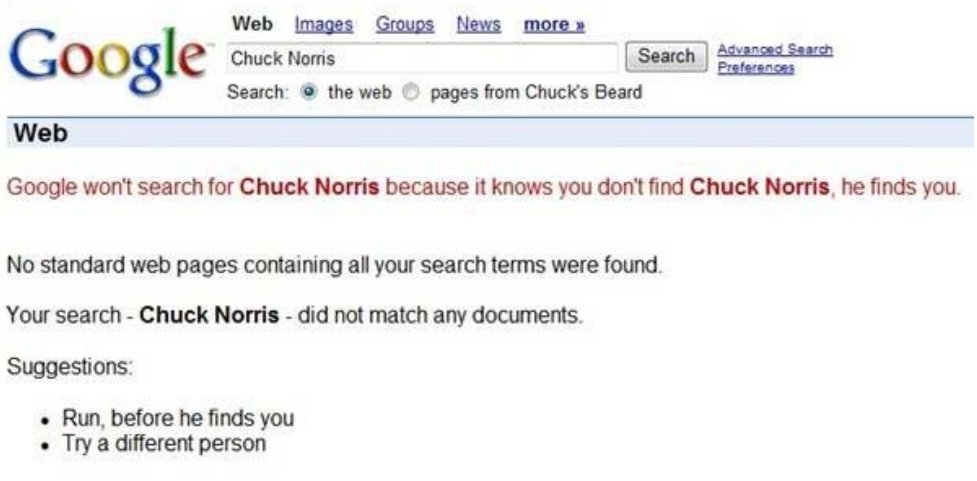
Liar -> Tony Blair

Google bombs



El caso de google bomb más conocida en español: ladrones -> SGAE

Google bombs



http://www.nochucknorris.com/

Esto no es un exactamente una “google bombs” . Cuando el usuario buscaba Chuck Norris en google y pulsaba el “voy a tener suerte” era redirigido a la página

www.nochucknorris.com que simula la SERP (search engine result page) de google con el mensaje:

“Google no ha querido buscar a Chuck Norris porque sabe que tu no encuentras a Chuck Norris, él te encuentra a tí”.

7. PAGERANK

Manning atribuye el origen de pagerank en 1965 + o –. Pero otros autores (wikipedia) autores atribuyen el nombre PageRank a Larry Page (cofundador de Google junto a Sergey Brin).
Lo que si parece claro es que se debe a Larry Page la forma eficiente de calcular el PageRank y su uso en los buscadores web.

Antecedentes del PageRank

Análisis de referencias en la bibliografía científica.

Una referencia dentro de un artículo científico a otro artículo puede verse como un enlace del primer artículo al segundo.

- 1ª Propuesta: Se puede medir la similitud entre dos artículos por el solapamiento de los artículos que los citan (cocitation similarity).

¿Tiene utilidad para los artículos científicos?

¿Tiene utilidad para los buscadores web?

El “cocitation similarity” se usa en bibliometría para comparar artículos científicos.
Si dos artículos son citados por los mismo se consideran que son similares.
Esto trasladado a la web nos permitiría a partir de una página encontrar páginas similares a esta, pero no es lo que necesitamos para encontrar

páginas en función de una consulta.

Antecedentes del PageRank

Análisis de referencias en la bibliografía científica.

- 2ª Propuesta: Las referencias a un artículo pueden considerarse como una medida del impacto de ese artículo.

¿Tiene utilidad para los artículos científicos?

¿Tiene utilidad para los buscadores web?

¿Todas las referencias deben valer lo mismo?

Las citas recibidas por un artículo individualmente o por una revista en su conjunto es una medida ampliamente utilizada para medir el IMPACTO (la relevancia) del artículo o la revista.

La medida más simple: por cada referencia recibida, el artículo obtiene un voto – no muy preciso.

Si trasladamos esta medida a la web, la equivalencia sería: el número de referencias -> número de enlaces de entrada a una página.
Pero, un sitio web con un número de enlaces de entrada elevando no NECESARIAMENTE es un sitios de alta calidad (spam de los enlaces).

Antecedentes del PageRank

Análisis de referencias en la bibliografía científica.

- 3ª Propuesta: Un artículo es de gran impacto si es referenciado por artículos de gran impacto.
  - En la web: una página es de gran relevancia si es indexada por páginas de gran relevancia.
  - El «peso» de cada enlace depende de la relevancia de la página que enlaza.

Para determinar la relevancia de una página se tiene en cuenta la relevancia de las páginas que la enlazan.

Esta es la base del PageRank.

¿Parece Circular? Puede ser formalizado de manera bien definida.

Un paseo aleatorio

Un internauta haciendo un paseo aleatorio por la web.

1. Comienza en una página cualquiera al azar.
2. En cada paso elige una nueva página aleatoriamente de manera equiprobable entre todos los enlaces salientes de la página donde está.
3. Repetir el paso 2 un número «infinito» de veces.

Un paseo aleatorio

Un internauta haciendo un paseo aleatorio por la web.

¿En qué pagina se encontrará el internauta?

¿Con qué probabilidad?

Cada página tendrá una probabilidad de que el internauta se encuentre en ella.

Las páginas «más importantes» tendrán una mayor probabilidad.

Intuitivamente: la probabilidad de que el internauta esté en una página es el PageRank de esa página.

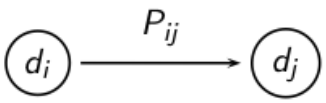
No se sabe en que página está el internauta pero en un estado estacionario habrá una distribución de probabilidad entre todas las páginas.

Esta probabilidad es el PageRank

El paseo aleatorio: una cadena de Markov

Una cadena de Markov viene definida por:

- Un conjunto de N estados.
- Una matriz N x N de probabilidad de transición, P.
- $\forall\ 1 \leq i,j \leq N, P_{ij}$ es la probabilidad de pasar al estado j estando en el estado i.
- $\forall\ 1 \leq i \leq N$, se debe cumplir que $\sum_{j=1}^{N} P_{ij} = 1$.



Vamos a realizar una formalización de paseo aleatorio usando para ello el concepto de cadena de Markov, el grafo de la web se interpreta como una cadena de Markov de forma que los estados son las páginas web.

En cada paso del paseo aleatorio estamos en una de las páginas.
Para **1 ≤ i, j ≥ N**, la entrada de la matriz $\mathbf{P_{ij}}$ nos dice la probabilidad de que **j** sea la página siguiente, dado que el internauta está en la página **i**.

El paseo aleatorio: una cadena de Markov

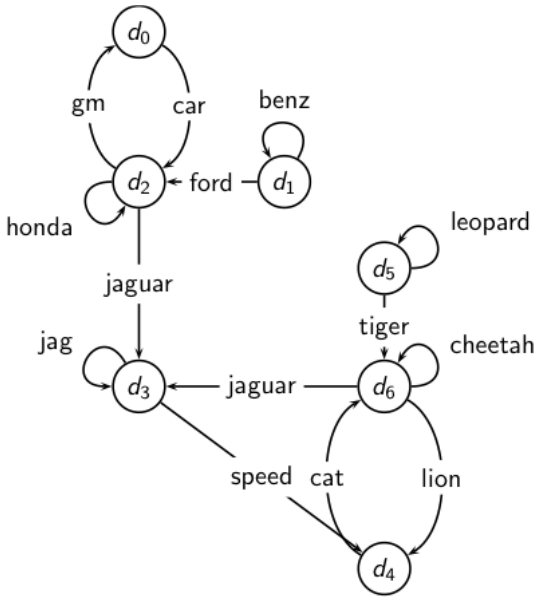
Un internauta haciendo un paseo aleatorio por la web visto como una cadena de Markov:

- Cada página web es un estado de la cadena.
- Sea ***out(i)*** el conjunto de todas las páginas alcanzables desde la página ***i***.
  - $\forall\ 1 \leq i \leq N, \quad \forall\ j \in out(i), \quad P_{ij} = \frac{1}{|out(i)|}$

Para este paseo aleatorio por la web, definimos la probabilidad $\mathbf{P_{ij}}$ de forma equiprobable como se indica en la fórmula de la diapositiva.

Ejemplo de paseo aleatorio

Grafo:



Sea el ejemplo de la diapositiva en el cual aparecen 7 nodos, numerados de $\mathbf{d_0}$ a $\mathbf{d_6}$.

Ejemplo de paseo aleatorio

Matriz de enlaces:

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Calculamos la matriz de enlaces del ejemplo.

Ejemplo de paseo aleatorio

Matriz de probabilidades de transición, ***P***

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Calculamos la matriz de probabilidades de transición del ejemplo siguiendo la defición anterior.

Tasa de visitas a largo plazo

La tasa de visitas a largo plazo de una página ***d*** es la probabilidad de que un internauta realizando un paseo aleatorio esté en la página ***d*** en un momento determinado.

El PageRank de una página es su tasa de visitas a largo plazo.

Teorema 1: Para una cadena de Markov ergódica hay una única tasa de visitas a largo plazo.

Consecuencia: para poder calcular el PageRank el grafo de la web debe ser una cadena de Markov ergódica.

Definición de la Tasa de visita a largo plazo.

¿Qué propiedades debe tener el grafo de la web para que la tasa de visitas a largo plazo pueda ser definida?

El gráfico de la web debe corresponder a una cadena de Markov ergódica.

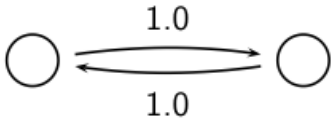
Primero un caso especial: el gráfico de la web no debe contener callejones sin salida.

Cadena de Markov ergódica

Una cadena de Markov es ergódica si es irreductible y aperiódica.

- Irreductible: Desde cualquier estado se puede alcanzar cualquier otro, no necesariamente en un solo salto.
- Aperiódica: No existe ningún estado tal que todos los caminos desde él a sí mismo tengan una longitud múltiplo de un periodo ***k*** > 1.

Ejemplo de cadena periódica de periodo k = 2



Definimos qué es una cadena de Markov ergódica.
El ejemplo es periódico porque cualquier camino de un estado a él mismo tiene longitud multiplo de 2: 2, 4, 6, 8.
No es necesario que pase en todos los estados para serlo.
Si ningún estado de la cadena es periódico -> la cadena es aperiódica.
Es nuestro caso esta propiedad no tiene demasiada importacia,
la solución que utilizaremos para solucionar el problema de la irreductibilidad hará que la cadena sea ergódica sin preocuparnos de la aperiodicidad.

Cadena de Markov ergódica

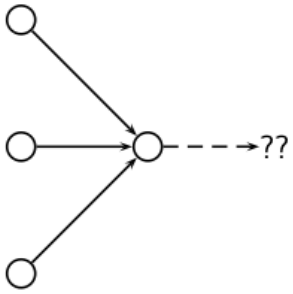
Una cadena de Markov es ergódica si es irreductible y aperiódica.

- Irreductible: Desde cualquier estado se puede alcanzar cualquier otro, no necesariamente en un solo salto.
- Aperiódica: No existe ningún estado tal que todos los caminos desde él a sí mismo tengan una longitud múltiplo de un periodo $k > 1$.

¿Es la web ergódica?

¿ES LA WEB UNA CADENA DE MARKOV ERGODICA?

Callejones sin salida



- La web está llena de callejones sin salida.
- El internauta aleatorio se quedará atrapado.
- Con callejones sin salida la web no es ergódica.
- Tasa de visitas a largo plazo no está bien definida.

El grafo de la web no es una cadena de Markov ergódica, está llena de callejone
s sin salida, es decir, páginas que no incluyen ningún enlace a otra página.
Se hace necesario definir una estrategia para convertir la matriz de probabilidades
de transición de una grafo web de forma que se corresponda con una cadena ergódica.

Solución a los callejones sin salida: teletransporte

- Si estamos en un callejón sin salida:
  - Saltar equiprobablemente a cualquier página web con probabilidad $\frac{1}{N}$
- Si no estamos en un callejón sin salida:
  - Con probabilidad $\propto$: Saltar a cualquier página web. Salto a cualquier página con probabilidad $\frac{\propto}{N}$
  - Con probabilidad $1 - \propto$: Saltar utilizando alguno de los enlaces de la página de forma equiprobable.

Definimos el teletransporte para resolver el problema.

Solución a los callejones sin salida: teletransporte

- $\propto$ es un parámetro ajustable. Típicamente $\propto = 0{,}1$.
- La probabilidad de salida en los callejones no depende de $\propto$. Sólo depende del número de páginas.

Ejemplo: 1000 páginas web ($N = 1000$) y $\propto = 0{,}1$.

- En los callejones sin salida elegimos cualquier página web con probabilidad *0,001*.
- En una página con 4 enlaces, podemos elegir:
  - 996 páginas con probabilidad *0,0001* (*0,1 / 1000*).
  - 4 páginas con probabilidad *0,2251* (*0,0001 + 0,9 / 4*).

Recordemos que para cualquier estado (página) la suma de las probabilidades de pasar a todas las páginas es 1.

Veamos un ejemplo de un grafo web con 1000 páginas y un parámetro alfa de 0,1.

Cadena de Markov ergódica

- Teorema 1: Para una cadena de Markov ergódica hay una única tasa de visitas a largo plazo.
- En el límite (un número grande de saltos) cada estado es visitado en proporción a esta tasa.
- No importa el estado de inicio del paseo.
- Esta tasa es la distribución de probabilidad en estado estacionario.

Recordemos el teorema que ya lo hemos visto antes.

Cadena de Markov ergódica

- La cadena de Markov definida por la web añadiéndole el teletransporte es una cadena de Markov ergódica.
  - Se puede ir a cualquier página desde cualquier página (Irreductibilidad).
  - No hay estados periódicos (aperiodicidad).
- La web con teletransporte tiene una distribución de probabilidad en estado estacionario.
- Esta probabilidad para cada página es su PageRank.

Vector de probabilidades para el paseo aleatorio

- El vector de probabilidades $\vec{x}$ $(x_1, x_2, \cdots, x_N)$ indica con que probabilidad está el paseo aleatorio en cada estado.
- El internauta aleatorio está en el estado i con probabilidad x_i.
- $\sum_{\forall i} x_i = 1$.

Ejemplo:

Al iniciar el paseo:

(0 0 0 … 1 … 0 0 0)

Posteriormente:

(0.05 0.01 0.02 … 0.2 … 0.01 0.05 0.03)

Definimos x como el vector de probabilidades que indica la probabilidad asociada a cada estado de la cadena de Markov de ser visitado en un tiempo dado en el paseo aleatorio. Su dimensión es N, el número de páginas web.

Evolución del vector de probabilidades

- En un instante de tiempo *t*, el vector de probabilidades es $\vec{x} = (x_1, x_2, \cdots, x_N)$.
- Dada la matriz de probabilidades de transición *P*, la probabilidad de pasar a la página j estando en la página i es P_{ij}.
- ¿Cuál será el valor del vector de probabilidades en el instante *t+1*?.
- La probabilidad de estar en el estado j en el instante *t+1* será: $\sum_{\forall i} x_i P_{ij}$.
- El vector de probabilidades en *t+1* se puede expresar como: $\vec{x}P$.

La probabilidad de estar en t+1 el estado j es:

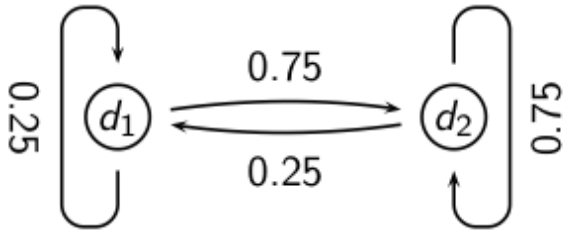
- la probabilidad de estar en el estado 1 en el tiempo t por la probabilidad de pasar del 1 a j
- + la probabilidad de estar en el estado 2 en el tiempo t por la probabilidad de pasar del estado 2 al j
- + …
- + probabilidad de estar en el estado N en el tiempo t por la probabilidad de pasar de N a j.

PageRank representado como vector

- El vector de probabilidades $\vec{\pi}$ $(\pi_1, \pi_2, \cdots, \pi_N)$ representa la distribución de probabilidad en estado estacionario. Es la evolución del vector $\vec{x}$ cuando el tiempo crece indefinidamente.
- π_i es la tasa de visitas a largo plazo del estado i.
- π_i es el PageRank para la página i.
- El PageRank se puede representar como un vector (muy grande) con un valor por cada página web.

Ejemplo de PageRank

¿Cuál es el PageRank para este ejemplo?



Veamos un ejemplo de una cadena de Markov ergódica con dos estados y las probabilidades de transición que se indicasn en la figura.

Ejemplo de PageRank

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.25$ $P_{21} = 0.25$	$P_{12} = 0.75$ $P_{22} = 0.75$
t_0	1	0	0.25	0.75
t_1	0.25	0.75	0.25	0.75
t_2	0.25	0.75	(converge)	

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

PageRank: $\vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$

Calcularemos el PageRank para el ejemplo, el vector de probabilidades en estado estacionario.

Las 3 filas de la izquierda de la tabla hacen referencia a los valores que va tomando el vector x en las diferentes iteraciones
Calculamos los valores del vector x para cada tiempo, empezando por t=0, asumiendo una inicialización del vector a los valores (1,0).
En la parte superior derecha hemos indicado la matriz de probabilidades de transición.
Y en la parte inferior derecha, los resultados de los nuevos cálculos para el vector x en las distintas iteraciones, valor que trasladamos a la parte correspondiente de la izquierda de la tabla.

En la parte inferior se muestran las fórmulas de cálculo para el tiempo t.
Y finalmente el resultado del PageRank, cuando se llega a la convergencia, y los valores del vector x no cambian al iterar.

Cálculo del PageRank ($\vec{\pi}$)

- Comenzar con cualquier distribución de probabilidad inicial $\vec{x}$.
- Después del primer paso estamos en $\vec{x}P$.
- Después del segundo paso estamos en $\vec{x}P^2$.
- Después de k pasos estamos en $\vec{x}P^k$.
- Método de las potencias: multiplicar $\vec{x}$ por potencias crecientes de P hasta que converja ($\vec{x}P^t = \vec{x}P^{t+1}$).
- Con el número suficiente de iteraciones, independientemente del valor inicial de $\vec{x}$ se obtiene el PageRank $\vec{\pi}$.

Lo que vamos a ver aquí es un método para calcular el PageRank.

No es la forma más eficiente de hacerlo.

En 1999 Larry Page y Sergey Brin formalizaron una forma viable de calcularlo basada en las propiedades de los vectors propios.

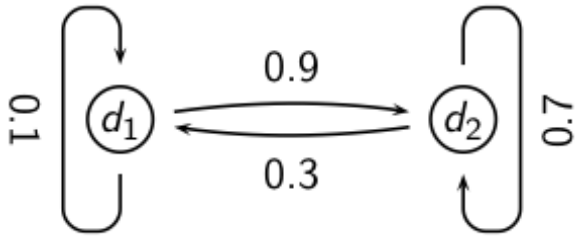
π es un vector propio de P: en el límite $\vec{\pi} = \vec{\pi}\,P$

Dadas las características de P, los chicos de google "inventaron" cómo hacer los cálculos viables para matrices tan tan grandes. Este método está basado en el método de las potencias.

El método de las potencias puede necesitar de MUCHAS iteraciones.

Ejemplo del método de las potencias

¿Cuál es el PageRank para este ejemplo?



Veamos otro ejemplo.

Ejemplo del método de las potencias

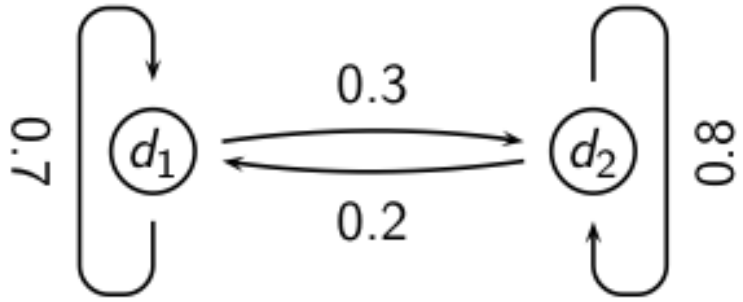
	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= x\vec{P}$
t_1	0.3	0.7	0.24	0.76	$= x\vec{P}^2$
t_2	0.24	0.76	0.252	0.748	$= x\vec{P}^3$
t_3	0.252	0.748	0.2496	0.7504	$= x\vec{P}^4$
			. . .		. . .
t_∞	0.25	0.75	0.25	0.75	$= x\vec{P}^\infty$

$P_t(d_1) = P_{t-1}(d_1) * 0.1 + P_{t-1}(d_2) * 0.3$
$P_t(d_2) = P_{t-1}(d_1) * 0.9 + P_{t-1}(d_2) * 0.7$

PageRank: $\vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$

El PageRank del documento 1 es 0.25 y el del documento 2 es 0.75.

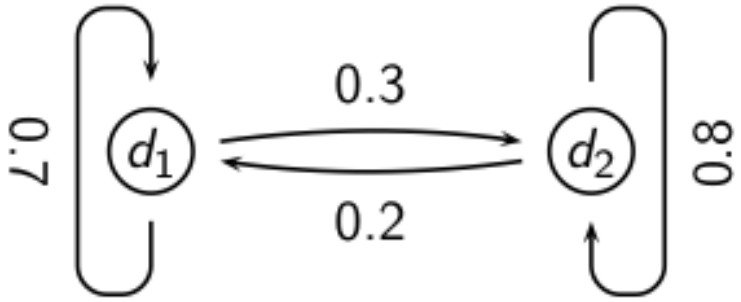
Ejercicio: Calcular el PageRank utilizando el método de las potencias



Nota: máximo 5 iteraciones

Se pide calcular el PageRank del grafo de la diapositiva, realizando un máximo de 5 iteraciones.

Solución



El PageRank del documento 1 es 0.4

El PageRank del documento 2 es 0.6

Solución del ejercicio.

Solución

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65	0.375	0.625
			. . .	
t_∞	0.4	0.6	0.4	0.6

Pasos del ejercicio.

Utilización del PageRank en la recuperación de información en la web

Para calcular el PageRank:

1. A partir de la matriz de enlaces construir la matriz de probabilidades de transición P_0.
2. Aplicar el teletransporte para obtener una nueva matriz de probabilidades de transición P.
3. Utilizando un vector de probabilidades inicial $\vec{x}$ y aplicando el método de las potencias obtener el vector $\vec{\pi}$.

$\vec{\pi}_i$ es el PageRank de la página representada por el estado i.

Vamos a ver cómo aplilcar el método estudiado para el cálculo del PageRank a partir de la matriz de enlaces de un grafo web.

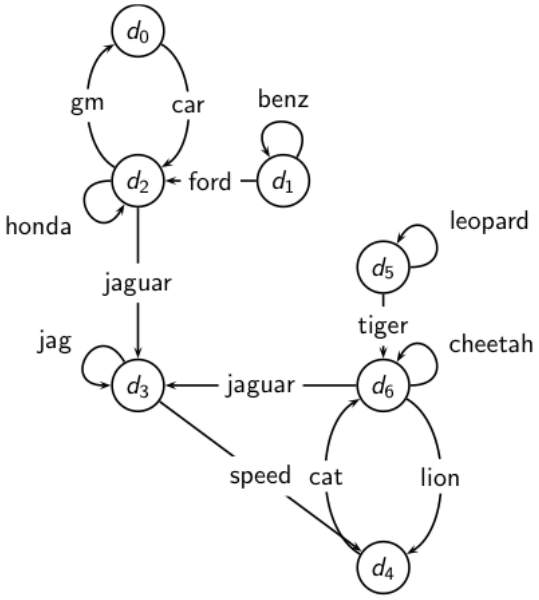
Utilización del PageRank en la recuperación de información en la web

Para responder a una consulta del usuario:

1. Recuperar todos los documentos (las páginas web) que satisfacen la consulta planteada por el usuario.
2. Utilizar el PageRank de cada página para ordenar la lista de resultados.
3. Presentarle al usuario la lista ordenada de páginas web.

Del grafo de páginas web al PageRank

El grafo:



Veamos un ejemplo, aquí partimos del grafo y vamos construyendo las matrices y las iteraciones para el cálculo del PageRank.

Del grafo de páginas web al PageRank

Matriz de enlaces:

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Del grafo de páginas web al PageRank

La matriz de probabilidades de transición inicial

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Del grafo de páginas web al PageRank

La matriz de probabilidades de transición con teletransporte ($\alpha = 0.14$)

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.02	0.02	0.88	0.02	0.02	0.02	0.02
d_1	0.02	0.45	0.45	0.02	0.02	0.02	0.02
d_2	0.31	0.02	0.31	0.31	0.02	0.02	0.02
d_3	0.02	0.02	0.02	0.45	0.45	0.02	0.02
d_4	0.02	0.02	0.02	0.02	0.02	0.02	0.88
d_5	0.02	0.02	0.02	0.02	0.02	0.45	0.45
d_6	0.02	0.02	0.02	0.31	0.31	0.02	0.31

Se ha utilizado un alfa=0,14 para el teletransporte.

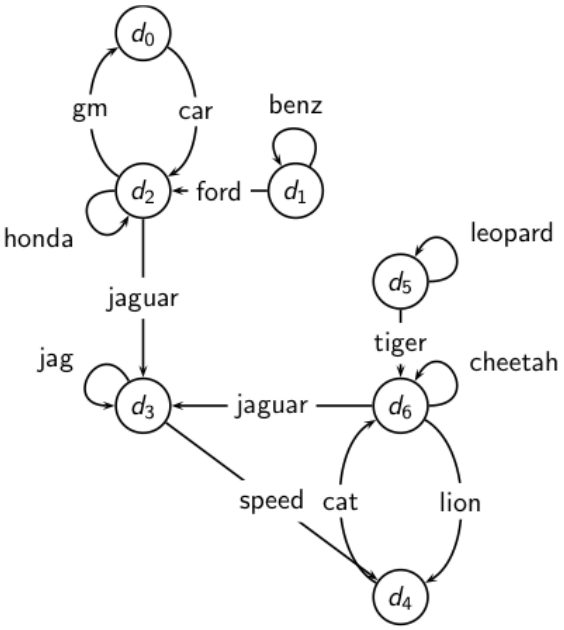
Del grafo de páginas web al PageRank

Aplicar el método de las potencias

	$\vec{x}$	$\vec{x}P^1$	$\vec{x}P^2$	$\vec{x}P^3$	$\vec{x}P^4$	$\vec{x}P^5$	$\vec{x}P^6$	$\vec{x}P^7$	$\vec{x}P^8$	$\vec{x}P^9$	$\vec{x}P^{10}$	$\vec{x}P^{11}$	$\vec{x}P^{12}$	$\vec{x}P^{13}$
d_0	0.14	0.06	0.09	0.07	0.07	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05
d_1	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_2	0.14	0.25	0.18	0.17	0.15	0.14	0.13	0.12	0.12	0.12	0.12	0.11	0.11	0.11
d_3	0.14	0.16	0.23	0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25
d_4	0.14	0.12	0.16	0.19	0.19	0.20	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
d_5	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_6	0.14	0.25	0.23	0.25	0.27	0.28	0.29	0.29	0.30	0.30	0.30	0.30	0.31	0.31

Al aplicar el método de las potencias se obtienen los resultados de la diapositiva apllicando 13 iteraciones.

Del grafo de páginas web al PageRank



PageRank	
d_0	0.05
d_1	0.04
d_2	0.11
d_3	0.25
d_4	0.21
d_5	0.04
d_6	0.31

Consideraciones sobre el PageRank

- En la actualidad no se utiliza sólo el PageRank para ordenar el resultado de una consulta en un buscador.
- Se utiliza una combinación ponderada de muchos factores: distancias entre página y la consulta, distancias entre el texto de los enlaces y la consulta, el PageRank de las páginas, situación geográfica, …
- El PageRank, una versión más evolucionada, sigue siendo importante.
- Es necesario detectar el link spam para que el PageRank sea más fiable.

No podemos saber cómo hacen la ordenación google, yahoo, bing ...

ES ALTO SECRETO.

Es la clave de su negocio -> que las páginas más útiles para el usuario salgan las primeras.

8. HITS: HUBS Y AUTHORITIES

Hyperlink-Induced Topic Search (HITS)

Ideado por Jon Kleinberg en su etapa en IBM.

A diferencia del PageRank, HITS se basa en la idea que
las páginas son importantes no solo si son enlazadas por
muchas (PageRank) sino también si ellas enlazan a muchas páginas relevantes.

HITS: Hyperlink Induced Topic Search

- Premisa en la que se basa HITS: hay dos tipos diferentes de relevancia en la web.
  - Hub. Es una página que contiene una buena lista de enlaces a páginas que contienen información útil.
  - Authority. Es una página que contiene respuesta directa a necesidades de información.
- PageRank no hace distinción entre estos dos tipos de relevancia.
- Muchas de las páginas que enlazan páginas tipo authority son hubs.

La idea detrás de Hubs and Authorities surgió de una visión particular de la creación de páginas web cuando Internet se estaba formando originalmente; es decir, ciertas páginas web, conocidas como hubs, sirvieron como directorios grandes que en realidad no tenían autoridad en la información que tenían, pero se usaron como compilaciones de un amplio catálogo de información que condujo a los usuarios directamente a otras páginas autorizadas. En otras palabras, un buen hub representa una página que apunta a muchas otras páginas, mientras que una buena autoridad representa una página que está vinculada por muchos hubs diferentes.

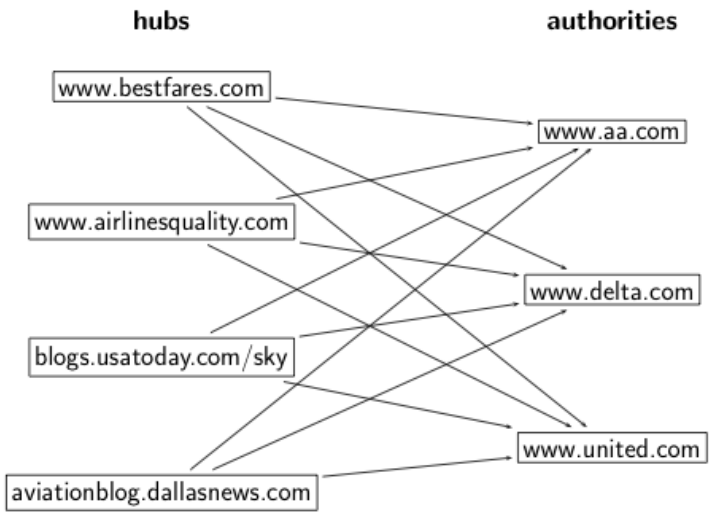
Hubs y Authorities

- Una buena página hub para un tema enlaza a muchas páginas authority para ese tema.
- Una buena página authority para un tema es enlazada por muchas páginas hub de ese tema.
- Es una definición circular que se puede transformar en un proceso iterativo.

Ahora desarrollamos un esquema en el cual, dada una consulta, se asigna a cada página web dos puntuaciones: uno como hub y el otro como autoridad. Para cualquier consulta, calculamos dos listas clasificadas de resultados en lugar de una.

La clasificación de una lista es inducida por las puntuaciones del hub y la de la otra por las puntuaciones de autoridad.

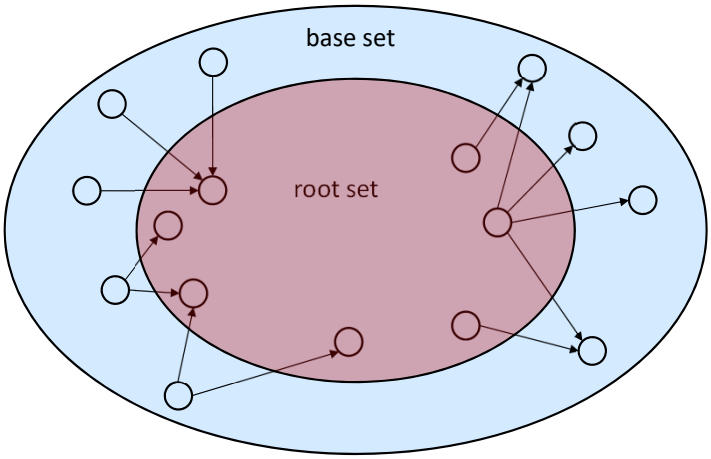
Ejemplo de Hubs y Authorities



Cálculo de hubs y authorities

- Se parte de una consulta inicial.
- Al conjunto de páginas que se obtienen como resultado de esa consulta se le llama conjunto raíz (root set).
- El conjunto inicial se amplia con todas las páginas que enlazan o son enlazadas a (o desde) él.
- Este conjunto ampliado de páginas se llama conjunto base (base set).
- El conjunto base puede representarse como un grafo.
- Los hubs y authorities se calculan sobre ese grafo.

Root set y base set



El root set tiene típicamente entre 200 y 1000 páginas mientras que el base set puede llegar a las 5000

Se siguen los enlaces salientes analizando las páginas en el conjunto raíz
Se encuentran los enlaces entrantes de d buscando todas las páginas que contengan un enlace a d.

Cálculo de hubs y authorities

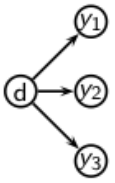
- Objetivo: Calcular para toda página *d* en el conjunto base una puntuación como hub (*h(d)*) y una puntuación como authority (*a(d)*).
- Inicialización: Para toda página *d*: *h(d) = 1, a(d) = 1*.
- Iterativamente recalcular los valores *h* y *a* de cada página hasta que converja.
- Resultado: dos listas con las páginas ordenadas atendiendo a las puntuaciones como hub y como authority.

Cálculo iterativo de ***h*** y ***a***

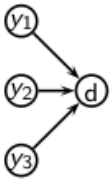
Repetir hasta que converja:

- Para cada documento ***d***:

$$h(d) = \sum_{d \to y} a(y)$$



$$a(d) = \sum_{y \to d} h(y)$$



d->y significa que en la página d hay un enlace a la página y.

Cálculo iterativo de ***h*** y ***a***

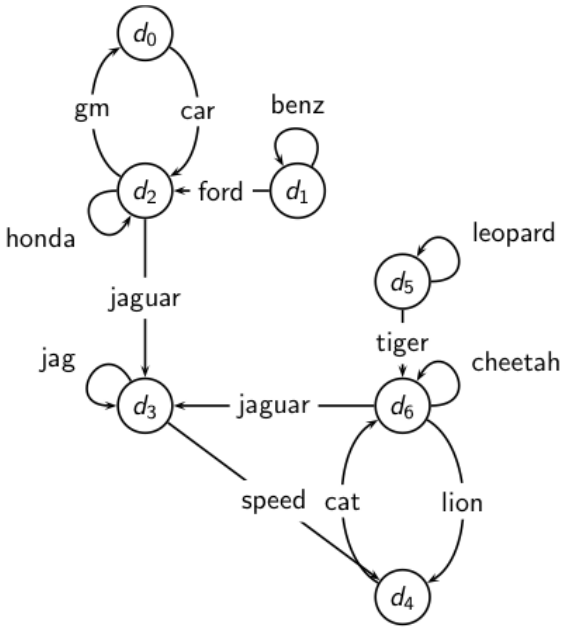
Una consideración adicional:

- Para evitar que los valores de ***h*** y ***a*** crezcan demasiado con cada iteración y
- Para asegurar la convergencia del algoritmo propuesto
  - Se deben escalar los valores de ***h*** y ***a*** después de cada iteración.

Independientemente del escalado, lo que interesa realmente es el orden relativo de las páginas atendiendo a la puntuación ***h*** y ***a***.

Este escalado es una especie de normalización

Ejemplo de cálculo de HITS (base set)



Sea el ejemplo de la figura.

Ejemplo de cálculo de HITS (base set)

Matriz de enlaces:

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

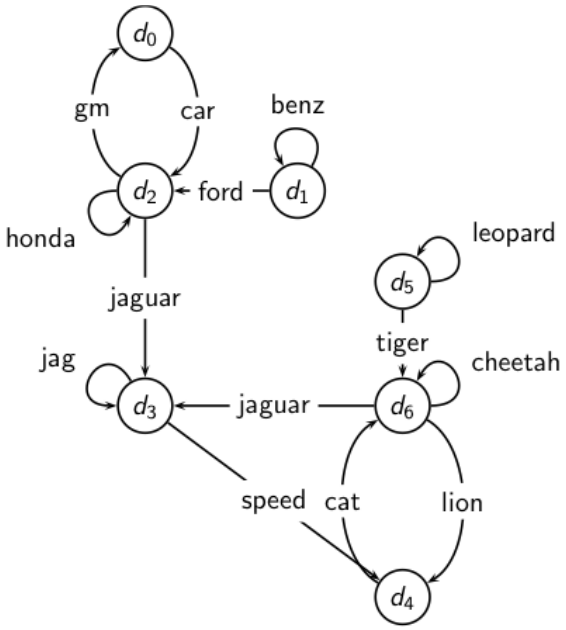
Sea esta su matriz de enlaces.

Ejemplo de cálculo de HITS (base set)

	Hub							Authority						
	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_0	d_1	d_2	d_3	d_4	d_5	d_6
t_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
t_1	1	2	3	2	1	2	3	1	1	3	3	2	1	3
t_2	3	4	7	5	3	4	8	3	2	6	8	5	2	6
t_3	6	8	17	13	6	8	19	7	4	14	20	13	4	15
t_4	14	18	41	33	15	19	48	17	8	31	49	32	8	33
t_5	31	39	97	81	33	41	114	41	18	73	122	81	19	82
Norm.	**0.07**	**0.09**	**0.22**	**0.19**	**0.08**	**0.09**	**0.26**	**0.09**	**0.04**	**0.17**	**0.28**	**0.19**	**0.04**	**0.19**
t_6	0.17	0.21	0.54	0.47	0.19	0.23	0.65	0.22	0.09	0.38	0.67	0.45	0.09	0.43
t_7	0.38	0.47	1.28	1.12	0.43	0.53	1.55	0.54	0.21	0.92	1.66	1.12	0.23	1.07
				...							...			
Norm.	**0.06**	**0.07**	**0.22**	**0.20**	**0.08**	**0.09**	**0.28**	**0.09**	**0.03**	**0.15**	**0.30**	**0.20**	**0.04**	**0.19**

En la diapositiva se muestran varias iteraciones de la aplicación
de las fórmulas para h(d) y a(d) de cada una de las páginas del ejemplo.
Después de la iteración para t=5 se ha llevado a cabo una normalización
de forma que la suma de h(d) para todo d es 1. Lo mismo para a(d).
Se sigue aplicando las fórmulas hasta un cierto número de iteraciones
y añadiendo alguna normalización adicional.

Ejemplo de cálculo de HITS (base set)



	Hub	Authority
d_0	0.06	0.09
d_1	0.07	0.03
d_2	0.22	0.15
d_3	0.20	0.30
d_4	0.08	0.20
d_5	0.09	0.04
d_6	0.28	0.19

Resultado para el ejemplo.

Comentarios sobre HITS

- HITS es capaz de encontrar buenas páginas independientemente de su contenido.
- A partir del conjunto base (que requiere de una consulta de usuario) sólo se hace análisis de enlaces, sin preocuparse por el contenido de las páginas.
- Muchas páginas del conjunto base pueden no contener ningún término de la consulta del usuario.
- En teoría, a partir de una consulta en un idioma se pueden recuperar páginas escritas en otros.
- El PageRank de cada página puede estar precalculado, HITS se debe calcular para cada consulta.

El peligro aquí es que las páginas recuperadas no tengan nada que ver con el tema de la pregunta.

El PageRank es independiente de la consulta.

El HITS es dependiente de la consulta

-> demasiado costoso para ser utilizado.