Recuperación de Información Multimedia

Descriptores de Texto (BoW, Tf-Idf, LSA, PageRank)

CC5213 – Recuperación de Información Multimedia

Departamento de Ciencias de la Computación
Universidad de Chile

Juan Manuel Barrios - https://juan.cl/mir/ - 2020



Modelos Estadísticos de Texto

- El lenguaje es una construcción compleja, depende del entorno y cambia en el tiempo
- El problema de codificar las reglas de gramática que definen el lenguaje es demasiado difícil para lograr buenos resultados
- En análisis de texto, usar modelos estadísticos simples sobre grandes cantidades de datos usualmente obtiene mejores resultados que implementar reglas complejas a mano considerando pocos datos

Ver Halevy, Norvig, Pereira, The unreasonable effectiveness of data. 2009. https://research.google.com/pubs/archive/35179.pdf



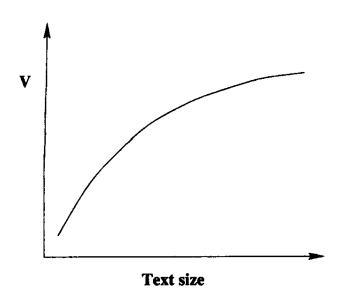
Modelo Bag-of-Words

- Un modelo se refiere al enfoque usado para representar documentos y calcular relevancias
- El modelo Bag-of-Words (BoW) representa documento mediante estadísticas de sus términos
- En principio, los términos corresponden a las palabras aisladas (se pierde la relación entre palabras)
- Requiere conocer el universo de términos posibles (vocabulario)



Tamaño del Vocabulario

- Ley de Heaps estima el tamaño del vocabulario (términos distintos) en el dataset
- Para n palabras, el tamaño del vocabulario usualmente es del orden Knβ
 - \square K y β depende del texto, usualmente K entre 10 y 100 y β entre 0.4 y 0.6



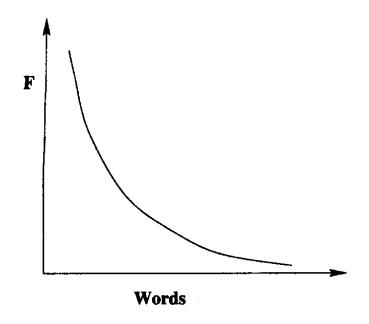


Frecuencias de Palabras

Ley de Zipf: la frecuencia de la *i*-ésima palabra más frecuente es:

 $f_i = f_1 \cdot \frac{1}{i^{\alpha}}$

- $\Box f_1$ es la frecuencia de la palabra más frecuente
- □ El valor de α depende del texto (entre 1.5 y 2.0)
- □ Es un tipo de "ley de potencia"



70

Descriptor de Texto

- Sea D={d₁, ..., d_n} el conjunto de todos los documentos en el dataset
- Sea K={k₁, ..., k_t} el conjunto de todos los términos distintos en el dataset (vocabulario)
- Para cada término k_i y documento d_j, se calcula un peso w_{ij} con la relevancia de k_i para d_j
 - □ Todos los w_{ij}≥0 y w_{ij}=0 ssi k_i no existe en d_i
 - □ Se define la función que calcula pesos g_i(d_i)=w_{ii}
- A cada documento se le asocia un vector:

$$d_j = (w_{1j}, ..., w_{tj})$$



Modelo Booleano



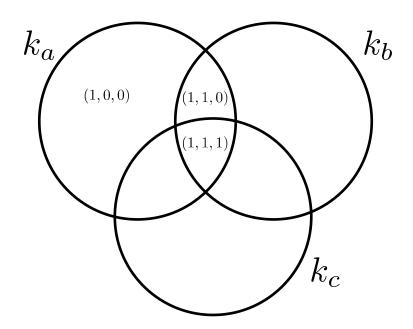
Modelo Booleano

- Modelo booleano considera que los términos están presentes o ausentes en un documento
 - □ Cada w_{ii} en {0, 1}
- Consultas se especifican como expresiones booleanas
 - □ Se consulta por la ocurrencia o no de cada término
 - □ Se pueden utilizar conectores lógicos: not, and, or
- Criterio de decisión binario, un documento es relevante o no lo es



Ejemplo

- Supongamos existen tres términos: $K = \{k_a, k_b, k_c\}$
- Se realiza la consulta: $q = k_a \wedge (k_b \vee \neg k_c)$
- La consulta se convierte a la forma normal disjunta (DNF): $q_{dnf} = (1,1,1) \lor (1,1,0) \lor (1,0,0)$





Consultas

- La consulta se resuelve comparando cada cláusula q_{cc} de q_{dnf} con cada documento
- Un documento es relevante si existe una cláusula de la consulta que coincidan todos sus pesos con los del documento

$$sim(q, d_j) = 1 \Leftrightarrow \exists q_{cc} \mid \forall i, \ w_i(d_j) = w_i(q_{cc})$$



Modelo Booleano

- Permite obtener todos los documentos que cumplen cierta condición
 - □ Bases de datos, Recuperación de datos
- Ventajas:
 - □ Formalismo claro y preciso
 - □ Simple
- Desventajas:
 - □ No existe un grado de similitud o relevancia, es decir, sim(q,d_i) es 1 o 0
 - La búsqueda usualmente recupera muy pocos o demasiados documentos



Modelo Vectorial



Modelo Vectorial

- Asignar pesos no binarios a los términos
 - Estos pesos se utilizan para determinar el grado de similitud entre documentos
- El modelo vectorial también toma en consideración documentos que sólo tienen un match parcial con la consulta
- Ranking: los documentos se priorizan de acuerdo a su grado de similitud con la consulta



TF: Term Frequency

- Representar las características que permiten agrupar documentos similares (similitud intracluster)
 - Representar los términos que aparecen en el documento
- Sea freq_{ij} la frecuencia del término k_i en el documento d_i, el factor tf está dado por:

$$tf_{ij} = \begin{cases} 1 + \log(freq_{ij}) & \text{si } freq_{ij} > 0\\ 0 & \text{si no} \end{cases}$$



IDF: Inverse Document Frequency

- Representar las características que permiten distinguir documentos que no son similares (disimilitud inter-cluster)
 - Términos que aparecen en muchos documentos no son útiles para distinguir documentos
- Sea N el número de documentos, n_i el número de documentos donde aparece el término k_i . La frecuencia de documento inversa para k_i es:

$$idf_i = \log\left(\frac{N}{n_i}\right)$$



Descriptor TF-IDF

Un documento d_j se representa por un vector de t dimensiones:

$$d_j = (w_{1j}, ..., w_{tj})$$

□ Donde w_{ij} es el peso del término i para el documento j, se calcula como:

$$w_{ij} = tf_{ij} \times idf_{i}$$

$$w_{ij} = \begin{cases} (1 + \log(freq_{ij})) \times \log(\frac{N}{n_{i}}) & \text{si } freq_{ij} > 0 \\ 0 & \text{si no} \end{cases}$$



Largo de Documentos

- La cantidad de palabras en un documento influye en el término tf
- Normalizar cada vector de pesos a largo unitario
 - Los vectores se ubicarán en la superficie de una hiper-esfera de radio 1

$$d'_{j} = (w_{1j}, ..., w_{tj}) ||d'_{j}|| = \sqrt{\sum_{i=1}^{t} w_{ij}^{2}} d_{j} = \frac{d'_{j}}{||d'_{j}||}$$



Función de Similitud

- El modelo vectorial propone una forma para medir el grado de similitud entre documentos
- Una función de similitud comúnmente usada es el coseno del ángulo entre vectores:

Producto escalar, producto interno o producto punto:

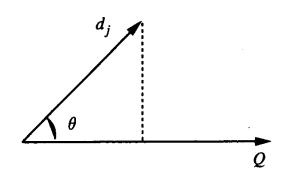
$$\vec{x} \cdot \vec{y} = ||\vec{x}|| \, ||\vec{y}|| \, \cos(\theta) \qquad \cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \, ||\vec{y}||}$$

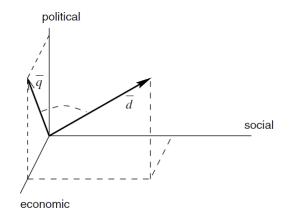
$$\cos(\theta) = \frac{\sum_{i=1}^{t} x_i y_i}{\sqrt{\sum_{i=1}^{t} x_i^2} \sqrt{\sum_{i=1}^{t} y_i^2}}$$



Similitud Coseno

La similitud coseno mide el parecido de dos documentos:





$$q = (w_{1q}, ..., w_{tq})$$

$$d_j = (w_{1j}, ..., w_{tj})$$

$$sim(q, d_j) = cos(\theta) = \sum_{i=1}^{t} w_{iq} w_{ij}$$

M

Función de Disimilitud

- El coseno del ángulo mide similitud:
 - □ 1 → vectores idénticos
 - \square 0 \rightarrow sin términos comunes
- Es posible convertir la similitud en disimilitud:

$$dis(q, d_j) = 1 - sim(q, d_j)$$

- □ 0 → vectores idénticos
- □ 1 → sin términos comunes
- Sin embargo no es una función de distancia apropiada porque no cumple la desigualdad triangular



Métricas

Se puede obtener una métrica usando el ángulo (distancia angular):

$$dis(q, d_j) = 1 - \frac{\arccos(sim(q, d_j))}{\pi}$$

Distancia euclidiana en función de similitud coseno:

$$dis(q,d_j) = \sqrt{2(1-sim(q,d_j))} \ = \sum_{i=1}^n (a_i-b_i)^2 = \sum_{i=1}^n (a_i-b_i)^2 = \sum_{i=1}^n a_i^2 + \sum_{i=1}^n b_i^2 - 2\sum_{i=1}^n a_ib_i$$



Modelo Vectorial

- Ventajas del modelo vectorial
 - □ El esquema de pesos mejora la eficacia
 - Se pueden recuperar documentos que se aproximen a la consulta
 - La medida de similitud permite rankear documentos
- Desventaja
 - Se asume que los términos son independientes



Generalización de Términos



Tokenizer

- Es la componente que divide la cadena de caracteres del documento original en sus términos o palabras
- Debe decidir qué caracteres componen una palabra y cuales son separadores:
 - □ ¿Se diferencian mayúsculas, minúsculas y acentos?
 - □ ¿Son símbolos como @ ; : _ ', parte de una palabra?
 - □ ¿Considerar números y fechas? ¿reemplazarlos por símbolos?
- Por ejemplo, una opción simple es que el Tokenizer use una expresión regular como: ([A-Za-z0-9_]+)

```
"El Sol salió a las 07:30"

→ [El, Sol, salió, a, las, 07, 30]

→ [el, sol, salio, a, las, #NUMERO#, #NUMERO#]
```



Stop Words

- Se llaman "Stop Words" a las palabras sin significado (conectores, preposiciones, artículos) que es posible no considerar al analizar el contenido de un documento
 - Usualmente son las palabras más frecuentes
- Se debe notar que:
 - □ La lista de stop words (stop-list) depende del idioma
 - Un término que aparece en todos los documentos no es útil para definir relevancia (sea o no una stop word)
 - □ Hay casos donde sí interesan todas las palabras



Stemming

- Considerar como término el inicio de cada palabra (word stem) que es compartido por todas sus variantes e inflexiones
 - □ No confundir con la raíz (root word) que se refiere al trozo relevante para su semántica (sin prefijos ni sufijos)
- Algoritmo de Porter es un método simple y muy usado
 - No produce necesariamente un resultado correcto (en el sentido lingüístico)

https://tartarus.org/martin/PorterStemmer/



Lemmatization

- Convertir cada palabra en su palabra base (lema) que corresponde a la palabra que aparece en un diccionario.
- El lema depende del significado de la palabra, el cual depende del contexto
 - La palabra "traje" puede referirse al verbo (lema "traer") o al sustantivo (lema "traje"). Se necesita el contexto de uso para decidir.
- Lemmatization puede llegar a ser bastante complejo en su implementación
 - Usualmente logra una mejora muy leve en efectividad con respecto a una técnica simple como stemming



WordNet

- Base de datos de palabras en inglés.
- Cuatro tipos de palabras: adjetivos, adverbios, sustantivos y verbos.
- Define grupos de palabras con igual significado (synsets)
- Cada synset tiene un código y una definición
- Define relaciones entre synsets
 - Mayor/Menor, Tiene/Es_parte
- Permite calcular distancia semántica entre palabras



N-Grams

- Considerar como términos en un documento las secuencias de n palabras consecutivas
 - □ En vez de palabras pueden ser secuencias de lemmas, stems, caracteres, etc.
- El tamaño del vocabulario (términos en los documentos) aumenta en forma significativa
 - Solo usar los n-grams que superan una cantidad mínima de repeticiones
- Modelo "bag-of-words" se convierte en un modelo "bag-of-n-grams"
- Notar que para usar n-grams o stems basta modificar el Tokenizer



N-Grams de Palabras

- Secuencias de n palabras consecutivas como un sólo término
 - □ También se incluyen secuencias de grado menor
- Por ejemplo:

el perro feroz	juega	con	la	pelota	roja	
----------------	-------	-----	----	--------	------	--

8 unigramas: el, perro, feroz, juega, con, la, pelota, roja

7(+8) bigramas: el perro, perro feroz, feroz juega, juega con, con la, ...

6(+15) trigramas: el perro feroz, perro feroz juega, feroz juega con, ...

5(+21) 4-gramas: el perro feroz juega, perro feroz juega con, ...



N-Grams de Caracteres

- Secuencias de n caracteres consecutivos
 - □ También incluir las secuencias de grado menor
- Naturalmente consideran word stems
- Por ejemplo:

```
e I perro y eI perrito juegan
```

```
char-unigramas: <u>e</u>, <u>l</u>, <u>,</u> <u>p</u>, <u>e</u>, <u>r</u>, <u>r</u>, <u>o</u>, <u>,</u> <u>y</u>, <u>,</u> <u>e</u>, <u>l</u>, <u>,</u> <u>p</u>, <u>e</u>, <u>r</u>, <u>r</u>, <u>i</u>, <u>t</u>, <u>o</u>, <u>,</u> <u>j</u>, <u>u</u>, ...
```

```
char-bigramas: <u>e-l</u>, <u>l-</u> , <u>-p</u>, <u>p-e</u>, <u>e-r</u>, <u>r-r</u>, <u>r-o</u>, <u>o-</u> , <u>-y</u>, <u>y-</u> , <u>e-l</u>, ...
```

char-4-gramas: <u>e-l--p</u>, <u>l--p-e</u>, <u>-p-e-r</u>, <u>p-e-r-r</u>, <u>e-r-r-o</u>, <u>r-r-o-</u>, <u>r-o--y</u>, ...



Ejercicio

Se tiene un dataset de 1.000 documentos con las siguientes estadísticas del vocabulario:

Palabra	Cantidad de veces que aparece en el dataset	Número de documentos en los que aparece
clavito	1.000	10
clavó	10	10
Pablito	10.000	1.000
qué	100	100
un	1.000	100

- Usando TF-IDF calcular la similitud entre las frases:
 - □ Documento 1: "Pablito clavó un clavito"
 - Documento 2: "¿Qué clavito clavó Pablito?"



Búsqueda

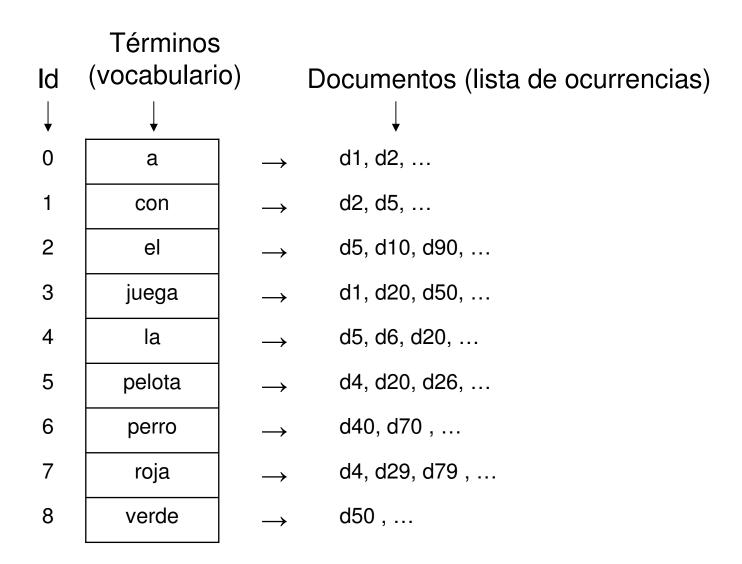


Índice Invertido

- El vocabulario contiene todos los términos en un dataset
- Las lista de ocurrencias de un término son todas las ubicaciones donde aparece en el dataset
 - Cada ocurrencia contiene (al menos) el id de documento
 - Dependiendo de la aplicación, se puede guardar más información como: posición absoluta dentro del documento (indexOf), número de palabra, número de página, número de párrafo, etc.
- El índice invertido es una tabla que asocia cada término con su lista de ocurrencias

M

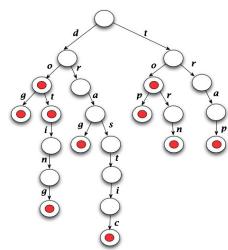
Índice Invertido





Búsqueda en el Vocabulario

- Para calcular cada descriptor es necesario un método eficiente para determinar el id en el vocabulario de cada término
 - Array ordenado de todo el vocabulario y usar búsqueda binaria: O(log(n))
 - ☐ Árbol de búsqueda
 - Trie=Árbol de prefijos: O(len(s))
 - Radix Tree o Patricia Tree comprimen caminos



M

Distancia entre palabras

- Distancia de Levenshtein (o distancia de edición)
 - \Box d(s,t) es el número mínimo de operaciones para transformar el string s en el string t
 - Cada operación de borrar, agregar o reemplazar un carácter tiene un costo fijo 1
 - □ Costo evaluación: O(n m)
 - □ Cumple las propiedades métricas
- Variantes:
 - Costo de inserción es mayor que costo de reemplazo
 - □ Costo de reemplazo variable: $c(n \rightarrow \tilde{n}) < c(n \rightarrow x)$
 - □ Operación de transposición: ab→ba



Requerimientos de Espacio

- El vocabulario usualmente requiere poco espacio
 - Se asume que es posible mantener el vocabulario en memoria
- Las listas de ocurrencias usualmente requieren mucho espacio
 - Almacenado contiguo de todas las ocurrencias (posting file)
 - Usar almacenamiento secundario y recuperar listas por palabra desde el disco
 - Algoritmos de compresión y búsqueda en texto comprimido



Búsqueda Eficiente

- La consulta se tokeniza, se obtienen sus términos y se calcula su vector tf-idf
- Con el índice invertido se obtienen las listas de ocurrencias que contienen los términos en la consulta
- Intersectar las listas de ocurrencias (seleccionar documentos con todos o casi todos los términos)
 - Para intersección eficiente las listas de ocurrencias deben estar ordenadas
 - □ Se puede realizar un merge lineal, o si una lista es mucho más corta (caso probable por la ley de Zipf) usar búsqueda binaria
- Los documentos seleccionados se ordenan según su similitud con la consulta (i.e., sim.coseno)





- Latent Semantic Analysis (LSA) o Latent Semantic Indexing (LSI)
- Para un vocabulario de t términos y un dataset de n documentos, sea A la matriz de descriptores:
 - □ A una matriz de txn con vectores (ej: tf o tf-idf) como columnas
 - \Box A^TA es una matriz de $n \times n$ contiene la similitud de documentos según términos comunes
 - □ AA^T es una matriz de txt contiene la similitud de términos según documentos comunes
- Idea: Eliminar términos correlacionados para reducir la cantidad de términos necesarios para representar cada documento del dataset
- Parecido a reducir dimensiones con PCA aunque operando directamente sobre A

M

- Usando el método Singular Value Decomposition se factoriza la matriz A=USV^T
 - \Box U matriz de $t \times r$
 - Contiene los vectores propios de AA^T
 - □ S matriz diagonal de r x r
 - Contiene la raíz cuadrada de los valores propios de AAT
 - Los *r* valores singulares $\lambda_1, ..., \lambda_r$ son distintos de cero
 - r ≤ min(t,n) se llama el rank de A
 - □ V matriz de *r* x *n*
 - Contiene los vectores propios de A^TA

M

- U y V son ortonormales:
 - $\square U^{\mathsf{T}}U = I$
 - $\square V^{\mathsf{T}}V = I$
- Se demuestra que:
 - $\square AA^{\mathsf{T}} = (\mathsf{USV})(\mathsf{USV})^{\mathsf{T}} = (\mathsf{USV})(\mathsf{V}^{\mathsf{T}}\mathsf{S}^{\mathsf{T}}\mathsf{U}^{\mathsf{T}}) = \mathsf{US}^{2}\mathsf{U}^{\mathsf{T}}$
 - $\square A^{\mathsf{T}}A = (\mathsf{USV})^{\mathsf{T}}(\mathsf{USV}) = (\mathsf{V}^{\mathsf{T}}\mathsf{S}^{\mathsf{T}}\mathsf{U}^{\mathsf{T}})(\mathsf{USV}) = \mathsf{V}^{\mathsf{T}}\mathsf{S}^{2}\mathsf{V}$



- Seleccionar los k valores singulares de mayor magnitud y reducir la dimensión de las matrices de r a k
 - □ A'=U' S' V'^T
 - □ U' de t x k representa el espacio de los conceptos
 - \square S'V'^T de $k \times n$ son los documentos en ese espacio

M

- Convertir los descriptores de documentos a su representación "latent semantic"
 - □ Usar V'^T o alternativamente multiplicar A^T por U'S'-1
- Para una consulta:
 - Calcular su descriptor q original (tf o tf-idf) de dimensión t
 - Obtener una representación de q que pueda ser comparada con los documentos
 - q'=q^TU'S'⁻¹
 - Comparar q' con cada documento usando similitud coseno



- Se reduce las dimensiones del vector, agrupando palabras similares
- Similitud entre palabras se refiere a palabras que tienden a aparecer simultáneamente
- A las nuevas dimensiones se les llama el espacio "semántico latente"
- Los documentos se representan por conceptos semánticos en vez de términos o palabras
- Lento de calcular (operaciones entre matrices grandes)





¿Cómo rankear la Web?

- La Web es enorme, con muchos elementos aleatorios, documentos que no son de confiar, spammers, etc.
 - □ ¿En quien confiar?
 - □ ¿Cual es la mejor respuesta cuando hay muchas fuentes de información?
- Necesitamos poder asignar un nivel de importancia a las páginas



Organizar la Web

- Primer intento: Directorios
 - Crear agrupaciones temáticas de sitios
- Dificultades:
 - Definir el universo de posibilidades del conocimiento y sus jerarquías
 - Ver Ontologías y Web Semántica
 - Asignar manualmante cada página o sitio a su categoría correspondiente





Organizar la Web

- Segundo intento: Buscadores
 - Usar Information Retrieval para localizar documentos relevantes en un conjunto de datos de confianza
- Dificultades:
 - Considerar alguna componente humana en la relevancia
 - El score no debiera depender únicamente de la calidad del match con la consulta
 - No todas las páginas web/sitios son igualmente importantes



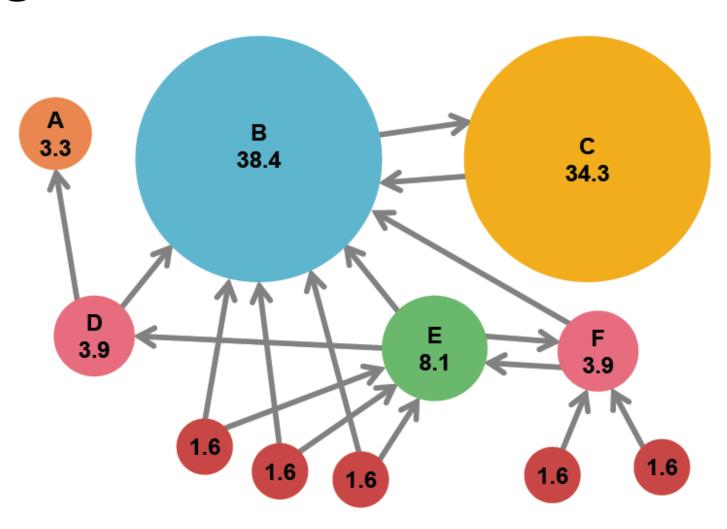
La Web como Grafo

- La Web puede ser vista como un Grafo:
 - Los nodos del grafo son las páginas
 - □ Las aristas del grafo son los hyperlinks
- Hay una gran diversidad en la conectividad de cada nodo
 - Rankear páginas según la estructura de sus conexiones



- Idea: Links como votos
 - Una pagina es más importante si tiene más links
- ¿Son todos los links iguales?
 - □ In-links vs Out-links
 - Usaremos In-links
- ¿Son todos los In-links iguales?
 - Los links (votos) dados por una página más importante debieran tener un peso mayor
 - Pregunta recursiva!

100





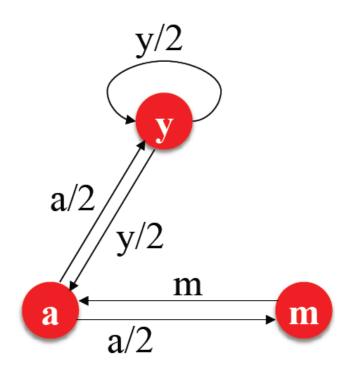
- Cada voto es proporcional a la importancia de su fuente
- La relevancia de una página es la suma de los votos de sus in-links
- Si una página con relevancia r tiene n out-links, cada link estará dando un voto de peso r/n
- El rank de la página j es:
 - □ i→j es un link de la página i a la página j
 - \Box d_i es el out-degree del nodo i (links de salida)

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$



Ejemplo

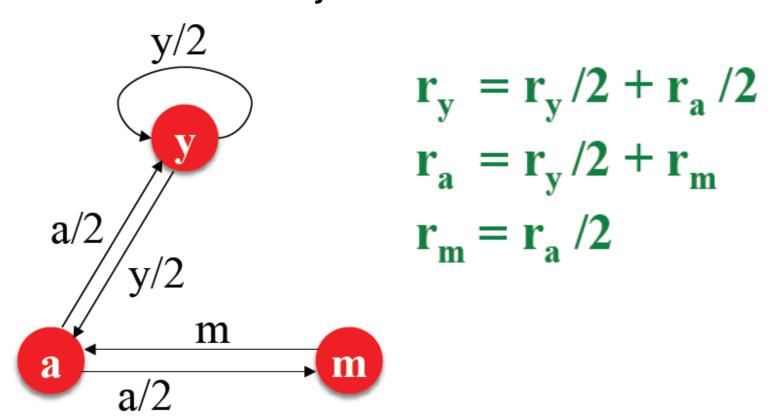
Escribir las ecuaciones para los scores de los nodos del siguiente grafo:





Ejemplo

Ecuaciones de flujo:



M

Ejemplo

- Si agregamos una restricción más:
 - □ La suma de los scores debe ser 1.

$$r_y + r_a + r_m = 1$$

$$r_y = r_y/2 + r_a/2$$

$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

■ Solución:
$$r_y = \frac{2}{5}$$
, $r_a = \frac{2}{5}$, $r_m = \frac{1}{5}$

70

Usando Matrices

- Escribir la propagación de votos en una matriz M
- Todos los links de salida de un nodo se escriben en una columna:
 - □ Por cada link i→j corresponde a 1/d_i en la celda M_{ii}
 - □ d_i es el grado de salida del nodo i
 - □ Las columnas de la matriz M deben sumar 1
- Las ecuaciones se pueden escribir como:

$$r = M \cdot r$$



Power Iteration

- Power iteration es un método para obtener el vector de scores r
 - □ Inicializar el vector r con valores iguales 1/n (todos inician con los mismos votos)

$$r^{(0)} = \left(\frac{1}{n}, \dots, \frac{1}{n}\right)^T$$

□ Iteración obteniendo nuevos valores de r

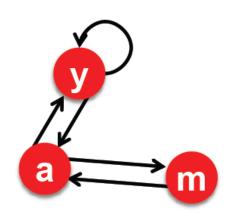
$$r^{(t+1)} = M \cdot r^{(t)}$$

□ Terminar cuando los scores converjan (la diferencia entre dos iteraciones es pequeña)

$$|r^{(t+1)} = r^{(t)}|_1 < \varepsilon$$



Ejemplo



	у	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$\begin{bmatrix} \mathbf{r}_{\mathbf{y}} \\ \mathbf{r}_{\mathbf{a}} \\ \mathbf{r}_{\mathbf{m}} \end{bmatrix} = \frac{1/3}{1/3} \quad \frac{1/3}{3/6} \quad \frac{5/12}{1/3} \quad \frac{9/24}{11/24} \quad \dots \quad \frac{6/15}{3/15}$$

Ver https://www.youtube.com/watch?v=CWzmnzB04eg



Random Walk

- PageRank puede ser interpretado como si un usuario de Internet esta navegando continuamente a través de los links de las páginas
 - □ Inicia en una página al azar (equiprobable)
 - Si en el tiempo t está en la página i, tomará alguno de los links i→j al azar (equiprobable) y en el tiempo t+1 estará en la página j
 - □ El proceso continua indefinidamente
- Luego de haber pasado mucho tiempo ¿Cuál es la probabilidad de estar visitando cada página en el tiempo t?



Random Walk

- Para que tenga sentido la interpretación Random Walk, toda página debe tener al menos un link de salida
 - Si no tiene salida, se crea una salida al azar para todas las páginas
 - □ Asegurar que todas las columnas de M suman 1
- Sin embargo pueden existir grupos de páginas sin salida dentro del grupo (spider traps)
 - □ Solución: Random Teleports
 - \square En cada tiempo t, existe una probabilidad β que el usuario siga un link y (1-β) que salte a una página al azar cualquiera



Formulación PageRank

La formulación completa es:

$$r_j = \beta \sum_{i \to j} \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

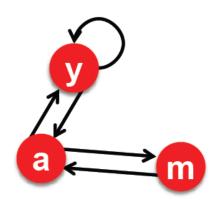
 En la práctica β es un parámetro fijado entre 0.8 y 0.9 (un usuario sigue unos 5 links antes de saltar al azar)



- En la práctica no hay que multiplicar las dos matrices
 - □ La matriz M es sparse
 - La matriz de random teleports no es necesario crearla
 - En cada iteración se lee la matriz M o se calcula el random teleport



Ejemplo



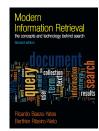
	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

Ver https://www.youtube.com/watch?v=CWzmnzB04eg



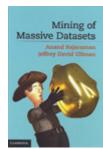
Bibliografía

- Modern Information Retrieval. Baeza-Yates, Ribeiro-Neto, 2011.
 - □ Cap. 3, 6, 8 y 9
- Multimedia Retrieval. Blanken, de Vries, Blok, Feng. 2007.
 - □ Cap. 3.5, 4.3 y 4.4.
- Information Retrieval. Algorithms and Heuristics. Second Edition. Grossman, Frieder. 2004
 - □ Cap 2.6 (LSA)
- Mining of Massive Datasets. Third Edition. Leskovec, Rajaraman, Ullman.
 - □ Cap 5.1 (PageRank)











Herramientas útiles

- Python, scikit-learn
 - http://scikit-learn.org/stable/modules/feature_extraction.html
- Java, Lucene
 - https://lucene.apache.org/core/
- R, package tm
 - http://tm.r-forge.r-project.org/