DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

Tirate un qué, tirate un ranking...

Métodos Numéricos
Segundo Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Aldasoro Agustina	86/13	agusaldasoro@gmail.com
Bouzón María Belén	128/13	belenbouzon@hotmail.com
Cairo Gustavo Juan	89/13	gjcairo@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Resumen

El resumen de no más de 200 palabras, deberá explicar brevemente el trabajo realizado y las conclusiones de los autores de manera que pueda ser útil por sí solo para dar una idea del contenido del trabajo.

Palabras claves

- Matriz Esparsa
- PageRank
- HITS
- In-deg

Índice

1. Introducción Teórica

A la hora de diseñar un Motor de Búsqueda hay varios aspectos a tener en cuenta, tales como: contar con acceso a las páginas disponibles en la red, tener una base de datos donde almacenarlas e indexarlas para su procesamiento posterior y ser capaces de ordenarlas de acuerdo a su importancia relativa dentro de dicha red. Nuestro trabajo se centrará en este último aspecto.

Existen varios métodos que priorizan distintas características de las relaciones entre las páginas para dar cierto orden a una red a partir de determinada búsqueda. Un criterio válido consiste en situar en una posición de mayor jerarquía a aquellas páginas que contengan mayor cantidad de coincidencias textuales con el concepto consultado. Éste podría no resultar óptimo en ciertos casos. Por ejemplo, si se buscara el string *"Red Social"*, intuitivamente se esperaría que entre los principales representantes de esta búsqueda se encontraran determinadas sitios web tales como Facebook o Twitter. Sin embargo, la cantidad de veces que estas páginas contienen al string *"Red Social"* puede no ser significativa y esto provocaría que no aparecieran en los primeros lugares las páginas genuinamente más vinculadas al concepto buscado.

Todos los métodos a desarrollar en este trabajo partirán del registro, la comparación y el análisis de los Links Salientes/Entrantes de la Red provista para ponderar el valor relativo de cada sitio en dicho sistema.

1.1. PageRank, HITS, In-deg

El trabajo consistirá en el estudio de distintos aspectos de los siguientes métodos: PageRank, HITS e In-deg. Los mismos se detallan a continuación:

PageRank *- Modelo del Navegante Aleatorio*

Este método consta de tres fases: exploración de la web y localización todas las páginas de acceso público; indexado de los datos desde el primer paso, de manera que se pueda acceder eficientemente a palabras claves o frases relavantes; y valoración de la importancia de cada una de las páginas en la base de datos. A nivel de nuestro desarrollo, sólo nos encargaremos de la última de las etapas mencionadas.

Teniendo un grafo dirigido, se le otorga a cada componente X_k del mismo un valor dado por la siguiente ecuación:

$$X_k = \sum_{j \epsilon L_k} \frac{X_j}{n_j}$$

Donde L_k es el conjunto de links entrantes a la página k y n_j es el número de links salientes desde la página j.

Luego, se construye una matriz A donde se encuentran -por filas- las respectivas ecuaciones para cada X_i, definidas como fue realizado anteriomente.

La resolución de este método se logra al hallar el autovector con autovalor asociado 1 para la matriz A. De acuerdo al trabajo de Bryan y Leise ACA PONER EL NUMERO DE REFERENCIA DE LA BIBLIOGRAFIA DONDE ESTE ESTE PAPER., este cálculo se computa mediante el método de la potencia.

Dicha matriz cuenta con ciertas mejoras que proporcionan ventajas en casos específicos. Por un lado, si alguna página p no tuviera ningún link saliente se considera que el navegante aleatorio saltará con equiprobabilidad a cualquiera de las otras páginas. De esta forma, se le otorga al vector columna p de la matriz A el valor de $\frac{1}{n}$ para cada componente. Por otro lado, existe un fenómeno denominado *"Teletransportación"* que considera la posibilidad de que dicho navegante se mueva de una página a otra pero no mediante los links existentes, sino tipeando la URL. Para modelar de manera óptima este suceso, se reemplaza a la matriz A por la matriz M definida bajo la siguiente ecuación: *M = (1-m)A + m.S* siendo m la probabilidad de que un navegante se *teletransporte* y S una matriz cuyos valores S_{ij} tienen todos el mismo valor: $\frac{1}{n}$ representando así una matriz donde la probabilidad de ir a cualquier página del grafo es uniforme.

El *Método de la Potencia* se realiza de manera iterativa, lo cual permite reducir el tiempo de cómputo para elevar a la k la matriz M. Si tenemos en cuenta el trabajo de Kamvar ACA PONER EL NUMERO DE REFERENCIA DE LA BIBLIOGRAFIA DONDE ESTE ESTE PAPER., presenta una herramienta de cálculo que permite encontrar el principal autovector de M en una serie menor de pasos modificando la ecuación

de la matriz M.

Hyperlink-Induced Topic Search (HITS)

El método planteado por Kleinberg ACA PONER EL NUMERO DE REFERENCIA DE LA BIBLIOGRAFIA DONDE ESTE ESTE PAPER. consiste en partir de una consulta sobre σ y focalizarse en una colección de páginas S_σ tal que sea relativamente pequeña, rica en páginas relevantes sobre el tema y contenga la mayoría de las autoridades más fuertes sobre el mismo. Considerando autoridad a una página que tiene la mayor cantidad de links entrantes provenientes de páginas vinculadas al tema, esto se realiza del siguiente modo:
a) Acorde a un parámetro *t*, se coleccionan las primeras t páginas rankeadas bajo una búsqueda basada estrictamente por texto. A este conjunto se lo denomina R_σ.
b) Se incrementa el conjunto R_σ añadiendo las páginas que tienen links entrantes y salientes al mismo, formando así el conjunto S_σ. Para cada página de R_σ se permite añadir, a lo sumo *d* páginas que la apunten y *d* páginas a las cuales apunte.
c) Se eliminan de S_σ los links intrínsecos, es decir no se tienen en cuenta links que apuntan a una página del mismo dominio que la página saliente.
d) Se admiten hasta *m* páginas del mismo dominio apuntar a cualquier página p. Esta idea no fue utilizada por el autor.

El conjunto obtenido hasta esta instancia lo llamamos G_σ. Nuestro trabajo asume un conjunto G_σ bien formado.

Se construye una matriz de adyacencia que denominaremos A, bajo la siguiente fórmula:

$$a_{ij} = \begin{cases} 1 & \exists \textit{ link desde i hasta j} \\ 0 & \textit{caso contrario} \end{cases}$$

A cada página i de la Web se le otorga un peso como Autoridad y un peso de Hub:
Peso de autoridad:

$$X_j = \sum_{i:i\to j} Y_i$$

Peso de Hub:

$$Y_i = \sum_{j:i\to j} X_j$$

Este algoritmo devuelve dos arreglos: uno representa los pesos de Hub y otro los pesos de Autoridad, teniendo una coordenada para cada página perteneciente al conjunto G_σ.

In-deg

Consiste en definir el ranking de las páginas utilizando sólo la cantidad de ejes entrantes a cada una de ellas, ordenándolos en forma decreciente.

2. Desarrollo

Deben explicarse los métodos numéricos que utilizaron y su aplicación al problema concreto involucrado en el trabajo práctico. Se deben mencionar los pasos que siguieron para implementar los algoritmos, las dificultades que fueron encontrando y la descripción de cómo las fueron resolviendo. Explicar también cómo fueron planteadas y realizadas las mediciones experimentales. Los ensayos fallidos, hipótesis y conjeturas equivocadas, experimentos y métodos malogrados deben figurar en esta sección, con una breve explicación de los motivos de estas fallas (en caso de ser conocidas).

2.1. Elección de las estructuras

Con el fin de elegir la estructura que representaría nuestra matriz esparsa, estudiamos tres tipos proporcionados por la cátedra: *Dictionary of Keys* (DOK), *Compressed Sparse Row* (CSR) y *Compressed Sparse Column* (CSC).

La primera consiste en un diccionario con doble clave (fila y columna) y su significado son los elementos de la matriz distintos de cero. De esta manera se saca provecho de la cantidad de elementos nulos de la matriz, garantizando una optimización en términos de espacio en memoria. Además esta implementación cuenta con la gran ventaja de que resulta simple construirla incrementalmente en un arreglo esparso y además puede ser traspuesta de manera sencilla (invirtiendo el orden de las claves). Sin embargo, el principal inconveniente reside en la necesidad de convertirla a otro formato para procesar los cálculos aritméticos. A causa de esto fue descartada la opción.

El modo de almacenamiento *Compressed Sparse Row* requiere la implementacion de tres arreglos (en nuestro caso vectores) que llamaremos val, ind_col y ptr_fila. El tamaño de los dos primeros está dado por la cantidad de elementos no nulos de la matriz. Mientras que el primero (val) almacena estos valores de izquierda a derecha y luego desde arriba hacia abajo, el segundo vector (ind_col) indica el número de columna para cada elemento. En otras palabras, el elemento almacenado en la posición i- ésima del vector ind_col representa la columna correspondiente al valor almanacenado en val_i. Por último el tercer vector (ptr_fila) tiene un tamaño equivalente a la cantidad de filas incrementada en uno, conteniendo los índices del comienzo de cada fila.

El modo de almacenamiento *Compressed Sparse Column* cuenta también con la implementación de tres arreglos llamados: val, ind_fila, ptr_col. El primero contiene todos los valores distintos de cero de la matriz, desde arriba hacia abajo y luego de izquierda a derecha. Ind_fila son los índices de fila correspondientes a dichos valores. Por último, ptr_col lista los índices donde comienza cada columna.

Por último, el tamaño del vector ptr_fila se encuentra determinado por la cantidad de filas incrementada en uno, y lista los índices que indican los valores de val que comienzan cada fila.

Ante a la disyuntiva acerca de cuál de estos últimos formatos escoger (CSR o CSC) decidimos realizar una serie de cálculos pequeños que nos permitieron notar que si nos situábamos en el formato de *Compressed Spare*, trasponer una matriz almacenada de manera CSC no sería más que interpretar sus mismos arreglos como CSC. Se incluye un ejemplo en *Apéndice C*. Fue decisión del grupo considerar el formato por defecto de la matriz el CSR (filas) y al trasponerlas sólo modificarle un bool que indique si está traspuesta y leerla y considerarla en adelante como CSC (columnas). Esta decisión fue tomada luego de que la cátedra nos confirmara que estaba permitido elegir una opción de las ofrecidas y adaptarla a nuestra provecho, siempre que se aclararan los cambios. Por este motivo, en el algoritmo de multiplicar una matriz por un vector se diferencia la manera en que la misma se encuentre almacenada y se obtiene el producto acorde a su formato. Se incluye el pseudocódigo de este algoritmo en la Subsección *Algoritmo multiplicación de una matriz por un vector*.

2.2. Algoritmo multiplicación de una matriz por un vector

Considerando la estructura elegida, nos vemos obligados a diferenciar dentro del algoritmo de multiplicación de una matriz por un vector de acuerdo al modo en que debe ser leído (CSR/CSC).

Para computar el cálculo de una matriz por un vector interpretándolo bajo la estructura *Compressed Sparse Row* se utilizó el siguiente algoritmo:

```
input  : Matriz m, Vector v
output: Vector res
for i ← 0 to cantidad de filas do
    inicio ← m.ptr_fil[i]
    fin ← m.ptr_fil[i+1]
    for j ← inicio to fin do
        col ← m.ind_col[j]
        res[i] ← res[i] + (m.val[j] * v[col])
    end
end
```

Se recorre el vector ptr_fil de la matriz, el cual indica en qué índice comienza cada fila. Para cada elemento de la fila actual, se asigna en el int col el número de columna correspondiente; y luego, se multiplica ese elemento con el correspondiente del vector v (v[columna actual]) y se suma en res[i], siendo i la fila actual.

Para computar el cálculo de una matriz por un vector leyéndolo bajo la estructura *Compressed Sparse Column* se utilizó el siguiente algoritmo:

```
input  : Matriz m, Vector v
output: Vector res
for i ← 0 to cantidad de filas do
    inicio ← m.ptr_fil[i]
    fin ← m.ptr_fil[i+1]
    for j ← inicio to fin do
        fil ← m.ind_col[j]
        res[col] ← res[fil] + (m.val[j] * v[i])
    end
end
```

Se recorre el vector ptr_fil de la matriz, el cual indica en qué índice comienza cada columna. Para cada elemento de la columna actual, se asigna en el int fil el número de fila correspondiente; y luego, se multiplica ese elemento con el correspondiente del vector v (v[fila Actual]) y se suma en res[fil].

2.3. Algoritmo de HITS

input : Matriz m, double tol
output: Vector x, Vector y

Inicializar vectores *x* e *y* con 1 en todas sus posiciones
Vector xp, yp
while *(No se haya llegado a la cantidad máxima de iteraciones i)* **do**
    m.$trasponer()$
    xp ← a.$multMatVect(y)$
    xp.$normalizar()$
    m.$trasponer()$
    yp ← a.$multMatVect(x)$
    yp.$normalizar()$
    if *(xp* $\simeq$ *x* $\wedge$ *yp* $\simeq$ *y)* **then**
        i ← Máxima Iteración
    else
        i++
    end
    x ← xp
    y ← yp
end
print *x* e *y*

La cantidad máxima de iteraciones la fijamos nosotros en 100.000
El $\simeq$ *considera la tolerancia (tol) pasada por parametro. Es decir, es equivalente a evaluar abs(x-xp)*$\leq$*tol*
Hay que poner algo sobre que copiamos el algoritmo del paper?

2.4. Algoritmo de PageRank

input : Matriz m, double c, double tol
output: x

Se inicializa el vector x en todos unos
Vector xp
while *(No se haya llegado a la cantidad máxima de iteraciones i)* **do**
    xp ← m.$MultMatVec(x)$
    for $i \leftarrow 0$ **to** $tamaño\ de\ xp$ **do**
        xp[i] ← xp[i] * c
        No sé cual forma poner ahora :(
        if *(xp ≃ x)* **then**
            i ← Máxima Iteración
        else
            i++
        end
        x ← xp
    end
end
normalizar el vector *x*
print *x*

La cantidad máxima de iteraciones la fijamos nosotros en 1.000.000
El ≃ considera la tolerancia (tol) pasada por parametro. Es decir, es equivalente a evaluar abs(x-xp)≤tol
Hay que poner algo sobre que copiamos el algoritmo del paper?

3. Resultados y discusión

Deben incluir los resultados de los experimentos, utilizando el formato más adecuado para su presentación. Deberán especificar claramente a qué experiencia corresponde cada resultado. No se incluirán aquí corridas de máquina.
Se incluirá aquí un análisis de los resultados obtenidos en la sección anterior (se analizará su validez, coherencia, etc.). Deben analizarse como mínimo los items pedidos en el enunciado. No es aceptable decir que los resultados fueron los esperados", sin hacer clara referencia a la teorica a la cual se ajustan. Además, se deben mencionar los resultados interesantes y los casos "patologicos.[en]contrados.

3.1. Convergencia de PageRank

Estudiar la convergencia de PageRank, analizando la evolución de la norma Manhattan (norma L1) entre dos iteraciones sucesivas. Comparar los resultados obtenidos para al menos dos instancias de tamaño mediano-grande, variando el valor de c. Opcional: Establecer una relación con la proporción entre lambda 1 = 1 y modulo de lambda 2.

3.2. Convergencia de HITS

2. Estudiar la convergencia de los vectores de peso x e y para HITS de forma similar al punto anterior.

3.3. Factor Temporal

3. Estudiar el tiempo de cómputo requerido por PageRank y HITS. Si bien ambos pueden se aplicados sobre una red genérica, cada algoritmo tiene un contexto particular de aplicación. Estudiar como impacta el factor temporal en este sentido.

3.4. Comparación de los tres Métodos

4. Estudiar cualitativamente los rankings obtenidos por los tres métodos. Para ello, se sugiere considerar distintos ejemplos de búsquedas de páginas web4. Analizar los resultados individualmente en una primera etapa, y luego realizar un análisis comparativo entre los tres rankings obtenidos.

3.5. Pequeños Ejemplos

5. Para cada algoritmo, proponer ejemplos de tamaño pequeño que ilustren el comportamiento esperado (puede ser utilizando las instancias provistas por la cátedra o generadas por el grupo).

4. Conclusiones

Esta sección debe contener las conclusiones generales del trabajo. Se deben mencionar las relaciones de la discusión sobre las que se tiene certeza, junto con comentarios y observaciones generales aplicables a todo el proceso. Mencionar también posibles extensiones a los métodos, experimentos que hayan quedado pendientes, etc.

5. Apéndices

5.1. Apéndice A

5.2. Apéndice B

Se adjunta aquí el algoritmo realizado para insertar, de a un elemento, los valores distintos de cero de una matriz en nuestra matriz esparsa:

input : Int fil, Int col, Double elem
output: Void

i ← índice donde comienza la fila fil pasada como parámetro
fin ← índice donde comienza la fila siguiente a la pasada como parámetro
Iterador itval ← crear iterador del vector val
Iterador itcol ← crear iterador del vector ind_col
while *(i<fin* $\wedge$ *col>itcol)* **do**
  Avanzar los dos iteradores
  i++
end
if *(i == tamaño del vector ind_col)* **then**
  insertar al final de ind_col el valor col pasado como parámetro
  insertar al final de val el valor elem pasado como parámetro
else
  insertar en la posición correspondiente al iterador el valor col pasado por parámetro en el vector ind_col
  insertar en la posición correspondiente al iterador el valor elem pasado por parámetro en el vector val
end
for $i \leftarrow fil + 1$ **to** $cantidadde filas$ **do**
  ptr_fil[i] ++
end

5.3. Apéndice C

Resultados que valga la pena mencionar en el trabajo pero que sean demasiado especificos para aparecer en el cuerpo principal del trabajo podrán mencionarse en sucesivos apéndices rotulados con las letras mayúsculas del alfabeto romano. Por ejemplo: la demostración de una propiedad que aplican para optimizar el algoritmo que programaron para resolver un problema.
- Va un ejemplo de que trasponer en csc es leerla como csr.

6. Referencias

Es importante incluir referencias a libros, artículos y páginas de Internet consultados durante el desarrollo del trabajo, haciendo referencia a estos materiales a lo largo del informe. Se deben citar también las comunicaciones personales con otros grupos.
PONER ACA LOS PAPERSSSSSSSSSSS