Métodos Numéricos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico 3

Grupo 14

Integrante	LU	Correo electrónico
Dabbah, Julián	015/09	djulius@gmail.com
Dahlquist, Ariel	383/10	ariel.dahlquist@gmail.com
Garrone, Javier	151/10	javier3653@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Introducción

El presente T.P. se preocupa por el algoritmo PageRank para medir la importancia de las páginas web resultantes de una búsqueda, en particular, en el enfoque provisto por el trabajo *Extrapolation methods for accelerating pagerank computations, Kamvar et al.*[1] Implementaremos dos de los algoritmos propuestos en esta publicación, siendo el segundo una optimización del primero, que resuelve el caso más general.

1.1. Algoritmo PageRank

Para ponderar la importancia de una páginas web, este algoritmo propone utilizar como medida la cantidad de enlaces que apuntan a esa página. Además, para agregar significación, cada enlace entrante a la página debe reflejar la importancia de la página desde la cual sale. Formalmente, el rank de la página k: x_k se define como, si $L = \{1 \dots n\}$ es el conjunto de páginas que apuntan a k y llamando n_j a la cantidad de links salientes de la página j:

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j}$$

Traduciendo esto matricialmente, podemos pensar en la matriz A donde $(A)_{i,j} = \frac{1}{n_j}$ si la página j apunta a i, $(A)_{i,j} = 0$ si no. Luego, el vector de ranks $X = (x_k)$ verifica $AX = X$, con lo cual, es un autovector de autovalor 1. Luego, en principio, para obtener los ranks bastaría con encontrar uno de éstos. Sin embargo, el modelo no representa adecuadamente el caso en que una página no apunte hacia otras: en ese caso su rank sería 0 y quedaría excluída. Este fenómeno aparece en la bibliografía como el problema de *dangling nodes*. También podemos justificar el modelo pensando en que la matriz representa una cadena de Markov, donde $(A)_{i,j}$ indica la probabilidad de, estando en la página (estado) j, pasar a la página (estado) i. Luego, podemos agregar las transiciones desde los *dangling nodes* uniformes como $\frac{1}{n}$, donde n es la cantidad total de páginas consideradas (es decir, en esta situación, se abandona la página pasando a cualquier otra con igual probabilidad). Sin embargo, esta solución trae problemas algebraicos, ya que no se puede asegurar que exista un autovector de autovalor 1 ni que el autoespacio asociado tenga dimensión 1. Si llamamos P a la matriz resultante, podemos solucionar el problema considerando una nueva matriz M que se obtiene como $M = cP + (1 - c)E$, para cualquier $0 < c < 1$, donde $E_{i,j} = \frac{1}{n}$. Así, M tiene efectivamente un autovector de autovalor 1 con multiplicidad 1.

1.2. Búsqueda de Autovalores

Como vimos, para encontrar el rank de las páginas es necesario encontrar un autovector de autovalor 1 de la matriz propuesta. Además, se puede asegurar que este autovalor siempre existe y es el autovalor principal, con lo cual desde el punto de vista numérico, cabe atacar este problema mediante el método de la potencia, es decir, calcular sucesivas iteraciones de $x^{(k+1)} = Ax^{(k)}$, a partir de algún x_0, en principio cualquiera. Sin embargo, dadas las dimensiones en las que se estaría trabajando, no cabe una implementación trivial. Para eso, se proponen las siguientes dos variantes.

1.2.1. Multiplicación Esparsa

Si bien la matriz original sobre la que trabajaría el algoritmo es típicamente esparsa (ya que la cantidad de enlaces entre dos páginas dadas es significativamente menor que la cantidad de páginas indexadas), las modificaciones para solucionar el problema de los *dangling nodes*, anulan esta propiedad. En esta situación, las sucesivas multiplicaciones de la matriz por el vector, resultarían muy costosas, tanto en tiempo como en espacio. Para esto, los autores proponen realizar esta modificación en los siguientes pasos, que como veremos, resulta equivalente a realizar la multiplicación directamente.

Por un lado, únicamente se almacena la matriz A como la definimos en primer momento, es decir, la que contiene la información sobre la distribución real de las páginas en la web y es altamente esparsa (lo cual permite ahorrar espacio de almacenamiento), y se modifica el vector resultante de realizar el producto. En concreto, se propone:

Algoritmo 1, *Kramver et al.*

$y = cAx$
$\omega = ||x||_1 - ||y||_1$
Devolver $y + \omega[\frac{1}{n}]_{\in \mathbb{R}^n}$

Es decir, se realiza la multiplicación aprovechando la matriz esparsa, y la corrección se agrega al final, en proporción a la diferencia de las normas entre el vector original y el resultado de la transformación. Dado que esta corrección altera a todos los coeficientes de toda la matriz original por igual, puede evitarse a la hora de realizar el primer producto y aplicarse al final, tomando como métrica la relación entre el vector original y el transformado.

[1] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating pagerank computations. In Proceedings of the 12th international conference on World Wide Web, WWW '03, pages 261–270, New York, NY, USA, 2003. ACM.

1.2.2. Extrapolación Cuadrática

Tal como lo explican los autores, la velocidad de convergencia del método de la potencia depende de la relación entre el autovalor principal y el subsiguiente. Luego, se propone agregar una modificación al algoritmo típico del método de la potencia para aprovechar las iteraciones anteriores utilizándolas para estimar una mejor aproximación del autovector buscado. Básicamente, se parte de la suposición de que cada iteración es combinación lineal de tres autovectores asociados a los tres únicos autovalores y se deducen relaciones entre ellos a partir de saber que $\chi_A(A) = O$ (Teorema de Hamilton-Cayley). Dado que $x^{(k+t)} = A^t x^{(k)}$, $(\chi_A(A))x^{(k)} = 0$ relaciona iteraciones sucesivas del método mediante los coeficientes del polinomio. A partir de éstos, entonces, se puede construir un sistema sobredimensionado para ajustar mediante cuadrados mínimos y apartir de los coeficientes obtenidos, construir el próximo vector para la iteración del método. Debido a que la suposición sobre la cantidad de autovectores no necesariamente es cierta, el vector obtenido no es la solución exacta (como se demuestra, en base a la suposición, en la publicación), pero de todas maneras, representa una buena aproximación, y, sobe todo, reduce notablemente los coeficientes con los que aparecen en la combinación lineal los autovectores sucesivos, acelerando la convergencia del método.

2. Resultados

El primer experimento se realizó utilizando una lista de 27 páginas web, las cuales devolvieron un total de 26 links (tan solo 12 de ellas dieron una cantidad positiva de links entrantes).

2.1. Propagación del error

El experimento fue realizado variando la constante c, la cual determina la importancia proporcional que se quiere entre la probabilidad de pasar a una página desde un link (c mas alto) contra la de escribir la url a mano (c mas bajo).

El criterio de parada es la diferencia de norma uno entre el autovector que se obtiene en cada iteración y el de la iteración anterior, el cual no debe ser menor que un epsilon de orden -15.

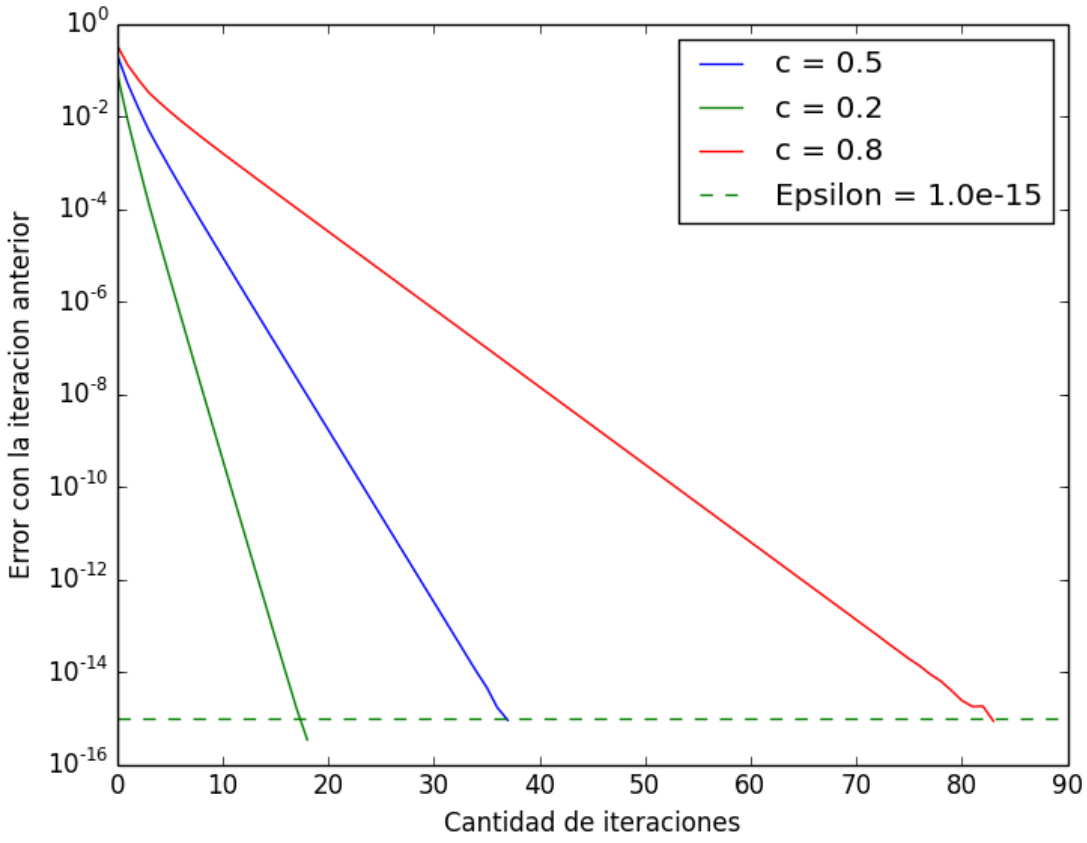

Figura 1

Se puede obvservar en la figura que cuanto menor es el c, más rápido converge el método. Creemos que dicho resultado se debe a que al darle menor importancia a los links directos, la matriz de probabilidades se encuentra mejor balanceada, por lo tanto los primeros autovalores de ella son mas parecidos y por ello se necesitan más iteraciones para lograr la convergencia. De hecho, se probó bajar mas el epsilon, el resultado fue que el método se estabiliza en pocos pasos más de los que muestra el gráfico, para cada línea, dejando un error igual a 0.

2.2. Relevancia de las páginas

La relevancia siguiente fue la dada por c = 0.5 (con los otros c cambiaban un poco el orden de las páginas del medio, pero las de mayor y menor relevancia se mantenán igual):

www.ole.com.ar	0.0676399
www.clarin.com	0.0622717
www.clasificados.clarin.com	0.0507299
www.ciudad.com.ar	0.0507299
www.lanacion.com.ar/	0.0474195
canchallena.lanacion.com.ar	0.0474195
www.rollingstone.com.ar	0.0442582
www.zonaprop.com.ar	0.0442582
www.hotmail.com	0.0440196
www.youtube.com	0.0426776
www.clarin.com/deportes	0.0362357
www.twitter.com	0.0355646
www.google.com	0.0284517
www.infobae.com	0.0284517
www.mamapuntocero.com.ar	0.0284517
www.pagina12.com.ar	0.0284517
www.yahoo.com	0.0284517
www.taringa.net	0.0284517
www.mercadolibre.com.ar	0.0284517
www.netbeans.com	0.0284517
www.github.com	0.0284517
www.assembla.com	0.0284517
www.gmail.com	0.0284517
www.9gag.com	0.0284517
www.twitter.com	0.0284517
maps.google.com.ar	0.0284517
www.stackoverflow.com	0.0284517

Tal como esperábamos ver, las 15 páginas que antes habíamos notado que no tenían links entrantes, son las últimas en la tabla, y no casualmente, tienen todas la misma relevancia.

Otra cosa que notamos al ir aumentando el c es que, dejando de lados los cambios en el orden de la tabla (que eran menores), los números cada vez se parecían más, ya que al darle mas importancia a las url que a los links, la cantidad de links de entrada que posee una página no entra tanto en juego y todas se consideran iguales en términos de relevancia.