CAIM - Sesión 5

Pagerank

Santiago Arxé i Carbona

Bryan Leonardo Salto Salao

29/11/2021

Implementación

En primer lugar se explicará cómo se ha implementado PageRank, cuál es su objetivo y cuáles han sido los principales problemas.

Para empezar, se han utilizado clases para representar las rutas y los aeropuertos con sus respectivos datos. Utilizando clases, se consigue guardar los datos de los dos ficheros .txt y poder acceder a ellos de manera fácil. Otra decisión que se ha tomado ha sido cómo representar el conjunto de aeropuertos y sus rutas. Para ello, tal y como recomendaba el enunciado se han utilizado *arrays* y hash o diccionarios (una mezcla). Gracias a esto, se ha podido acceder de la manera más óptima a los aeropuertos y rutas, sin desaprovechar espacio como se haría con una matriz.

A la hora de implementar el algoritmo de PageRank, se han presentado diversos problemas. El primero de todos, tan trivial como básico, fue entender todos los aspectos y variables que jugarían un papel principal en dicha implementación.

Como se comenta en el enunciado, uno de los pilares del pseudocódigo es el término $\frac{1-L}{n}$ que se suma en cada posición del vector y en cada iteración del bucle. Dicho término representa la probabilidad de saltar a un *link* aleatorio desde cualquier otra página del ecosistema. Sabiendo que para calcular el valor de PageRank de una página se deben sumar sus pesos de entrada, tiene lógica que también deba sumarse el peso de llegar a esa página desde páginas que no estén vinculadas de manera directa a ella.

Este mismo principio es el que ha entrado en juego a la hora de gestionar qué hacer con los aeropuertos de los que no salían vuelos. Dichos aeropuertos son problemáticos para cualquier red de transporte (o cualquier grafo dirigido en general) ya que actúan como callejones sin salida, o pozos (*sinks*) y echan por tierra muchas de las asunciones. Para gestionarlos, se ha trabajado con un parámetro relativamente sencillo: $\frac{L}{n}$, el cual representa la probabilidad de seguir un *link* en la página actual. Multiplicando ese valor por la cantidad de *sinks* que existen y dividiendo en el número de aeropuertos, se encuentra la probabilidad de terminar en un *sink*. Esta probabilidad también debe sumarse, ya que si en una iteración se termina en un *sink*, la probabilidad de llegar a los otros aeropuertos sería cero, algo que no tiene cabida en PageRank. De este modo, debe sumarse esa probabilidad de manera equitativa a todos los aeropuertos con tal de "normalizar" el comportamiento de los *sinks*.

Como también se comentaba en el enunciado, también debía escogerse el mecanismo (condición) de parada. Para un algoritmo de este tipo, una de las condiciones que siempre tiene más sentido es la de convergencia. Cuando dos iteraciones sean relativamente iguales (con un margen escogido y comentado más adelante), se para la ejecución.

Experimentación

Tal y como se comenta en el enunciado los valores preferidos al momento de calcular PageRank son $0.8 \leq L \leq 0.9$. Con tal de escoger el valor más óptimo tanto para el resultado como para el coste de ejecución, se han realizado unos cuantos experimentos que se resumen en la siguiente tabla.

L	Iteraciones	Tiempo(s)	máximo	mínimo
0.1	9	0.1482	0.00077 (DEN)	0.000164 (GLI)
0.2	12	0.2171	0.00135 (DEN)	0.000152 (GLI)
0.4	22	0.3768	0.00253 (DEN)	0.000126 (GLI)
0.8	87	1.4377	0.00515 (DEN)	5.29661e-05 (GLI)
0.85	119	1.9092	0.00557 (ORD)	4.10593e-05 (GLI)
0.9	184	3.0324	0.00620 (LAX)	2.83246e-05 (GLI)
0.95	377	6.3592	0.00689 (ORD)	1.46724e-05 (GLI)
0.98	956	16.2787	0.00732 (ORD)	5.9986e-06 (GLI)

Tabla 1: Resultados con un *stopping_threshold* de 1e-12.

Como se puede observar en la primera parte de la tabla 1, el número de iteraciones es bastante pequeño, al igual que el tiempo, por lo que no exige un coste muy grande. Por otra parte, la diferencia entre **mínimo** y **máximo** va aumentando conforme la L va creciendo, con lo que el algoritmo es capaz de clasificar cada vez mejor las páginas.

En el caso de L = 0.85, se logra un número de iteraciones suficientemente grande como para clasificar bien los nodos (aeropuertos) con un tiempo de computación aún muy pequeño (menos de 2 segundos). Dentro del intervalo [0.8, 0.9] se obtienen resultados bastante buenos sin sacrificar el tiempo excesivamente.

Cuando L > 0.9, el número de iteraciones aumenta mucho, y de su mano también lo hace el tiempo. En el caso L = 0.95 ya se obtiene un tiempo bastante alto de 6 segundos pero en L = 0.98 casi se triplica. La diferencia entre mínimo y máximo sigue aumentado pero cada vez menos por lo que no sale a cuenta estos valores de L para una pequeña mejora y un gran coste.

Para acabar, como observación se puede ver como hay mayor variación en el PageRank con una L más pequeña que con una más grande. Aunque las páginas importantes salen en la mayoría de los casos tomados aunque el número de iteraciones no sea tan grande.

Para acabar, cabe analizar el método de parada escogido, en este caso *convergence*. Como se ha comentado anteriormente, para parar la ejecución se utilizará un parámetro denominado *stopping_threshold*. La diferencia entre los valores del vector en la iteración *i* y los valores en la iteración *i + 1* deberá ser menor al *stopping_threshold* para que la ejecución termine. Para escoger un buen valor para *stopping_threshold*, se han realizado diferentes pruebas que se pueden ver en la tabla 2.

stopping_threshold	Iteraciones	Tiempo(s)
1e-4	7	0.1186
1e-5	20	0.3300
1e-10	91	1.4860
1e-15	162	2.6010
1e-20	No Termina Ejecución	No acaba

Tabla 2: Número de iteraciones y tiempo para diferente svalores de *stopping_threshold*.

Como se puede observar, entre [1e-4, 1e-10] el número de iteraciones es pequeño al igual que el tiempo. Entre [1e-10, 1e-15] el tiempo y las iteraciones aumentan considerablemente aunque todavía es razonable. Y para acabar, entre [1e-15, 1e-20] se aprecia como con 1e-20 el programa tarda tanto que no tiene sentido esperar a que termine la ejecución. Lo peor para un usuario impaciente. La recomendación final sería escoger un valor del segundo intervalo, por ejemplo 1e-12.