CAIM LAB5: PageRank

Jiajian Cheng
Yimin Pan

November 2019

1 Introducción

En esta práctica aplicaremos el algoritmo de `Pagerank` para una red de aeropuertos y rutas, que cuyo datos son extraidos desde OpenFlights y leidos por el programa en formato txt.

El algoritmo de `Pagerank` calcula la importancia de un nodo (documento) basándose en las aristas que entra en él y la importancia del nodo origen.

$$P_{k+1}[i] = \lambda \cdot \sum_j \frac{P_k[j] \cdot w(j,i)}{out(j)} + \frac{(1-\lambda)}{n},\ \forall (j,i) \epsilon Routes$$

- w(i,j) es el peso de la arista del nodo i a j, en nuestro caso es el número de repeticiones de la misma arista.
- out(i) representa el número de aristas que salen del nodo i.
- λ es el factor damping.
- $\frac{1-\lambda}{n}$ es el factor de teleportación, es decir probabilidad de un salto aleatorio.

2 Implementación

Al inicio, el vector P está inicializado todos a $\frac{1}{n}$, de este modo asegurando siempre $\sum_i P[i] = 1$. En cada iteración actualizamos el pagerank de cada nodo, y como tenemos implementado, el número de iteraciones dependerá de una toleracia. Definida como el máximo de diferencia entre los elementos de P_k y P_{k+1} (en el código tenemos un vector Q que hace de auxiliar), paramos cuando:

$$\forall i \epsilon P_k \ \ \forall j \epsilon P_{k+1} \ \ \mid P[i] - P[j] \mid \leq tol$$

Cada vez que se hace menor la diferencia entre iteraciones, sabemos que estamos convergiendo a hacia una la solución. Ya que en realidad estamos calculando una aproximación del pagerank teórico.

En cuanto a estructura de datos, para cada aeropuerto a_i tenemos un diccionario de rutas (edges) entrantes a a_i y un índice que indica la posición que ocupa dicho aeropuerto en la lista de aeropuertos. De este modo reducimos la complejidad en respecto a crear una matriz $n \cdot n$, $O(n \cdot m)$ vs $O(n^2)$, siendo m el máximo de aristas que entra en un nodo, por lo tanto $m << n$.
También nos hemos ahorrado la lista y diccionario de rutas que había en la plantilla, dado que ahora la información de las rutas están incluidos en el diccionario que hay en la clase aeropuerto (la lista que hay dentro de la clase también lo hemos eliminado).

3 El problema de los nodos desconexos

En general no hemos encontrado gran dificultad a la hora de implementar *Pagerank*, era simplemente seguir el pseudocódigo del enunciado.
Pero al hacer una iteración nos dimos cuenta que el sumatorio sobre P no se acercaba a 1. Esto se debe a que había nodos que no tenían aristas entrantes, o incluso ni salientes. Pero dado el factor damping $(1 - \lambda)$, sabemos que podemos eventualmente llegar a estos nodos. De este modo, su pagerank obviamente no es 0 y tiene aportación sobre los demás nodos.
La solucón sería añadir aristas virtuales desde estos nodos a todos los $n - 1$ demás, sin embargo este cambio hará que los $O(n)$ aristas se convierta en $O(n^2)$ y consecuentemente un incremento importante en tiempo y espacio. En realidad podemos calcular la aportación de los nodos desconexos sin tener que modificar el grafo.

- Por la forma de calcular el pagerank, sabemos que existe una parte que no varia de iteracón en iteración. Todos los nodos i que no tienen aristas de salida, si los conectamos con aristas virtuales a todos los demas nodos, tenemos $out(i) = n - 1$. Por otra parte, el factor λ tampoco varia. Por lo que si el número de nodos de este tipo es m, tenemos $m \cdot \frac{\lambda}{n-1}$. En el codigo lo llamamos `disconnectedPRfixed`.

- La parte que puede variar es el sumatorio sobre $P_k[j] \cdot w(j, i)$, pero como no existe repeticiones para estas aristas, sabemos que $w(j, i) = 1$, por lo tanto solamente tenemos que determinar cuánto vale $\sum P_k[j]$. En el código lo llamamos `disconnectPRvariable`. Para la primera iteración está claro que vale $\frac{1}{n}$ ya que es el valor que hemos inicializado el vector P. Después lo calculamos como:

  $$disconnectPRvariable_{k+1} = disconnectPRvariable_k \cdot disconnectedPRfixed + \frac{1 - \lambda}{n}.$$

  Es decir, la aportación total de los nodos desconexos (no tienen otras aristas entrantes que las virtuales) más el factor de teleportación.

De este modo, para calcular el pagerank de los aeropuertos conexos, solo tenemos que sumarle $disconnectPRvariable \cdot disconnectedPRfixed$.

4 Experimentación

En cuanto a la condición de parada, al principio por facilidad y para comprobar que el algoritmo funcionara ($\sum P = 1$), hemos fijado el número de iteraciones. Pero no seguiamos ningún criterio para elegir ese número. Por lo cual lo cambiamos por similitud de valores entre interaciones como habíamos descrito más arriba (tol). Para $\lambda = 0.85$, si poniamos valores grandes (10^{-5}), paramos con 20 iteraciones. Y si lo cambiamos a (10^{-16}) el número de iteraciones ya sube a 176, con muy poca variación sobre P en las últimas iteraciones. Por este motivo escogemos un valor más intermedio, que parase con alrededor de 100 iteraciones, esto es $tol = 10^{-12}$.

Fijado tol, si cambiamos λ de 0.85 a 0.3, podemos observar que ahora paramos con tan solo 17 iteraciones. Esto se debe a que, si λ es mayor, entonces $\frac{1-\lambda}{n}$ será mayor, y este factor de teleportación es igual para todos los nodos. Por lo tanto, este cambio hace que la diferencia de pagerank entre nodos será menor (un decimal de diferencia entre el mayor y menor), de allí menos iteraciones. Sin embargo, este factor no lo podemos ignorar, ya que nos impide entrar en un callejón sin salida al navegar sobre el grafo, y la posibilidad de llegar a nodos desconexos. Por lo que un número pequeño pero no casi nulo para $1 - \lambda$ como por ejemplo 0.15, es adecuado. De este modo, hemos elejido $\lambda = 0.85$, consiguiendo una diferencia de 4 decimales entre el mayor pagerank y el menor.