Sesión 5: Pagerank

Miguel Alcón Doganoc
Javier Lopez Calderon

16 de noviembre de 2017

1 Dificultades en la implementación

Hemos tenido varios problemas a la hora de implementar el algoritmo de *pagerank*.

1. Decidir y entender las estructuras necesarias para computarlo.
2. Al principio no tuvimos en cuenta que las rutas pueden incluir un aeropuerto de origen o destino que no tenemos en `airportList` y a la hora de acceder a algún aeropuerto a partir del origen o destino de una ruta dejaba de funcionar.
3. Cuando tuvimos el *pagerank* implementado nos dimos cuenta de que la suma de éste no daba 1, sino 0.6, porqué no controlábamos la contribución de los valores nulos (`outweight == 0`).

2 Selección de parámetros

2.1. Damping Factor (L)

Hemos probado con L = {0.8, 0.85, 0.9} y hemos obtenido que aunque varía el orden de los aeropuertos (según su *pagerank*), no lo hace por mucho. Por ejemplo, el primer aeropuerto en L = 0.9 pasa a ser el segundo con L = 0.85 y el tercero con L = 0.8. A parte de esto, el número de iteraciones y el tiempo de ejecución aumenta con L. Para L = {0.8, 0.85, 0.9} tenemos #iteraciones = {133,177,272} y tiempo = {4.658,6.372,9.602} (en segundos).

Hemos observado que un incremento de L tiende, en general, a aumentar la precisión del resultado con un importante coste de iteraciones y tiempo. Por este motivo elegimos L = 0.85 al ser un término medio.

2.2. Stopping Condition (n_decimals)

Nuestra función `stoppingCondition` compara los n decimales del *pagerank* de la iteración actual con el de la anterior y retorna True cuando son iguales (hasta que converge). Hemos probado para n = {2,4,8,16} y nos da #iteraciones = {0,23,77,177} y tiempo = {0.008,0.865,2.571,6.133}. A parte de esto, al comparar los resultados hemos observado que entre n = {2 y 4} tenemos 3322 diferencias (el orden de los aeropuertos difiere en 3322 posiciones), entre n = {4 y 8} tenemos 664 diferencias y entre n = {8,16} tenemos 4.

Seleccionamos n = 8 porqué entre n = {8 y 16} tenemos solo 4 diferencias y tardamos 3.562s y 100 iteraciones menos.