Ivan Cala Mesa
Grup 13

Informe lab 5: Implementando PageRank

El siguiente informe de CAIM trata sobre la aplicación del algoritmo PageRank en un contexto distinto al de la web informática: un grafo de rutas de vuelos. Para ello, trabajaremos con los datos proporcionados por OpenFlights, los cuales nos dan información relativa a aeropuertos y rutas a estos.
Trabajaremos fundamentalmente con solo aquellos aeropuertos con código IATA. La IATA es una de las principales instituciones para reglamentar y estandarizar el mundo de la aviación, por lo tanto es más apropiado, *for the sake of correction* limitarnos a estos. Además, son estos códigos los que nos ayudarán a cruzar los datos entre datasets.

Problemas

Posiblemente uno de los mayores problemas que he tenido se corresponde a elementos de la sintaxis y gramática de Python combinado con el hecho de que trabajamos con archivos externos, que se rigen bajo su propia sintaxis, y hay que filtrar.
Por ejemplo, idear cuál es la mejor estructura de datos para cada caso. ¿Estructuras asociativas tipo árbol? No sirve para todo, con lo que dije: hashmaps? Dado que trabajamos con una cantidad de datos masiva y queremos evitar costes agregados en construcción y búsqueda. Vale, pero entonces para depende qué tareas tendremos que extraerlos por fuerza bruta e idear una pipeline para los datos para por ejemplo, ordenarlos y realizar otro tipo de operaciones de este cariz. En sí mismo considero que tiene más complicación encontrar las técnicas de programación para lenguajes débilmente tipados como Python que tener las ideas de diseño.

Adicionalmente tuve problemas con el cómputo de las iteraciones, porque no modificaban el resultado y daba un vector igual. Era un problema de construcción de las propias clases Edge y Airport.

Por último, no he conseguido alcanzar la suma de Pageranks a 1. Creo que puede tener que ver con los aeropuertos sin ejes en el grafo, pero no he podido hacer nada más relativo a eso.

Comentarios y observaciones

Los detalles que plantean mayores matices en las implementaciones de PageRank son principalmente la condición de detención (cómo saber cuándo ya no es necesario seguir haciendo iteraciones) y el damping factor.

Relativo a la condición de detención, en un onliner he hecho que compare que las diferencias individuales entre iteraciones sean menores que un dado número muy

pequeño. Cuando se cumpla para todos los casos, ya no hará más iteraciones. Este número es cuestión de ir cambiando según las necesidades de precisión, pero hay un factor muy importante y es que la precisión máxima de los floating points en python es de 18 cifras, lo que hace que al poner números como por ejemplo, 10^(-20), el programa se quede ejecutando constantemente por no encontrar un resultado que jamás obtendrá. Para más precisión, haría falta modificar la implementación utilizando Decimals, y no Floats.
Por otro lado, la modificación del Damping Factor tiene efectos significativos sobre la convergencia.
En concreto, a menor sea su valor, más rápida será la convergencia en términos de iteraciones. Asimismo, esta rápida convergencia viene de la mano de que los valores que nos encontramos son más distantes entre sí cuánto más grande sea el damping factor, y menos cuando es menor.

Por ejemplo, manteniendo el resto de parámetros constantes, para un Damping factor de 0.2 una posible iteración hizo 16 iteraciones.

```
#Iterations: 16
Time of computePageRanks(): 0.2516329288482666
```

Para uno de 0.9 en mismas condiciones, 231.

```
#Iterations: 231
Time of computePageRanks(): 3.4013218879699707
```

Y básicamente en cuanto al tiempo, va casi directamente proporcional al número de operaciones.