Session 5: Page Rank

Q1 2023/2024

Àlex Domínguez Rodríguez

Mario Fernández Simón

Introducción

En esta práctica estudiaremos el cálculo del *PageRank* a través de un conjunto de datos de aeropuertos y rutas aéreas.

Estructuras de datos

Para representar los conjuntos de datos usamos 4 estructuras:

- *edgeList*: es una lista que guarda todos las rutas discriminación
- *airportList*: es una lista que guarda todos los aeropuertos que tengan código de identificación *IATA code*. Los aeropuertos tienen un parámetro ruta, donde guardamos todos los aeropuertos a los que este tiene una ruta.
- *edgeHash*: es un diccionario que guarda las rutas usando como clave el aeropuerto destino. Eso lo hacemos para que sea más fácil iterar entre las rutas de llegada al aeropuerto en el cálculo del *PageRank*. Durante la inicialización de esta estructura, nos aseguramos de que si una ruta la ya tenemos guardada, no volverla a guardar y, en consecuencia, incrementar en uno el peso de esa ruta. No se guardan en esta estructura las rutas que tengan aeropuertos que no estén dentro de nuestra lista de aeropuertos.
- *airportHash*: es un diccionario que guarda los aeropuertos por su código IATA. Se usa la misma discriminación que en *airportList*. Usamos esta estructura para tener actualizados los valores del peso de salida de los aeropuertos y para hacer cualquier consulta pertinente a los aeropuertos.

Para calcular el *PageRank* utilizamos dos diccionarios, *P* i *Q*. Los dos usan como key el código IATA de los aeropuertos.

P: Representa el valor actual de *PageRank* para cada aeropuerto, al comienzo de cada iteración. Durante el proceso de iteración, estos valores se actualizan gradualmente.
Q: Representa el valor calculado de *PageRank* para cada aeropuerto en la siguiente iteración. Los nuevos valores calculados se almacenan en *Q*.

En el caso de que la suma de las diferencias entre *P* i *Q*, término a término, sea menos de un número estipulado, nosotros usamos 1e-8 , damos por concluido el cálculo.

Explicación del algoritmo

1. **Inicialización de parámetros:**
    - *L* es el factor de amortiguación (damping factor).
    - *n* es el número total de aeropuertos.
    - *n_zeroOutDegree* es el número de aeropuertos que tienen un peso de salida igual a cero.
    - *aux* se inicializa con *1/n*, se utiliza como un término auxiliar en los cálculos.
    - *numberOuts* representa la contribución de los aeropuertos con un peso de salida igual a cero.
2. **Iteraciones:**
    - El código realiza un bucle de iteraciones *(max_iterations = 1000)* para calcular el *PageRank*.
    - Dentro de cada iteración, se inicializa un diccionario `Q` para almacenar los nuevos valores de *PageRank*.
3. **Distribución de *PageRank*:**
    - Se itera sobre los aeropuertos y se distribuye el *PageRank* desde los nodos con un peso de salida mayor que cero.
    - Se utiliza la fórmula de *PageRank* para distribuir el *PageRank* desde los nodos de origen a los nodos de destino a lo largo de los bordes ponderados.
    - Se aplica el factor de amortiguación *L* en el proceso de distribución.
4. **Efecto del Factor de Amortiguación:**
    - Se agrega el efecto del factor de amortiguación *(1 - L)/n* a todos los nodos.
    - También se ajusta el término auxiliar *aux* en cada iteración.
5. **Convergencia:**
    - Se verifica la convergencia comparando los valores actuales de *PageRank* (*Q*) con los valores anteriores (*Q*) utilizando una condición de parada basada en la diferencia absoluta (*epsilon*).
6. **Actualización y retorno:**
    - Se actualiza el valor de *PageRank P* con los nuevos valores calculados *Q*.
    - Se devuelve el número de iteraciones (*it*), los nuevos valores de *PageRank* (*P*), y el total restante del *PageRank* (*remaining_pagerank_total*), para saber la diferencia entre el la suma de los valores i 1, que es el valor ideal de la suma.

7. **Notas adicionales:**

El término *aux* se utiliza como un factor auxiliar en los cálculos durante el proceso de iteración del algoritmo de *PageRank*. Su papel principal es ajustar el valor del factor *(1-L)/n* en cada iteración. Se inicializa con el valor *1.0/n* y luego se utiliza en la actualización de la variable *Q[i.code]* en el bucle de iteración. Observemos cómo se utiliza:

```
Q[i.code] += (1 - L) / len(airportHash) + aux*numberOuts
```

Aquí, *aux* se multiplica por *numberOuts* y se suma al término *(1-L)/len(airportHash)*. La razón detrás de este uso es ajustar la contribución de los aeropuertos con un peso de salida igual a cero (*numberOuts*) en cada iteración.

Resultados

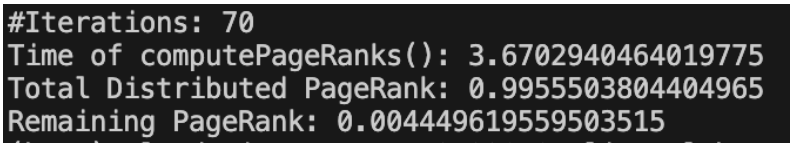

Figura 1: Sortida PageRank.py

Con el código descrito anteriormente, con un dumping factor de 0.85, hemos conseguido que el cálculo converja a las 70 iteraciones, en 3.67 segundos. Vemos que el error en la suma de los *PageRanks* es del 0.4%, las pequeñas discrepancias en el valor total pueden ser aceptables debido a la naturaleza iterativa y numérica del algoritmo, que trata con números con muchos decimales.

Experimentación

Para estudiar cómo se comporta el número de iteraciones, el tiempo de ejecución y el *PageRank* que falta por repartir según el valor de Dumping Factor. Hemos usado valores desde 0.1 a 0.95, incrementado de 0.05 en 0.05. Los valores los hemos conseguido haciendo 10 ejecuciones con cada valor.

Se pueden consultar los valores en el documento *Experimentacion_pagerank.csv* y el código usado para hacer la experimentación en *Experimentacion_pagerank.py*.

Experimentació iteracions

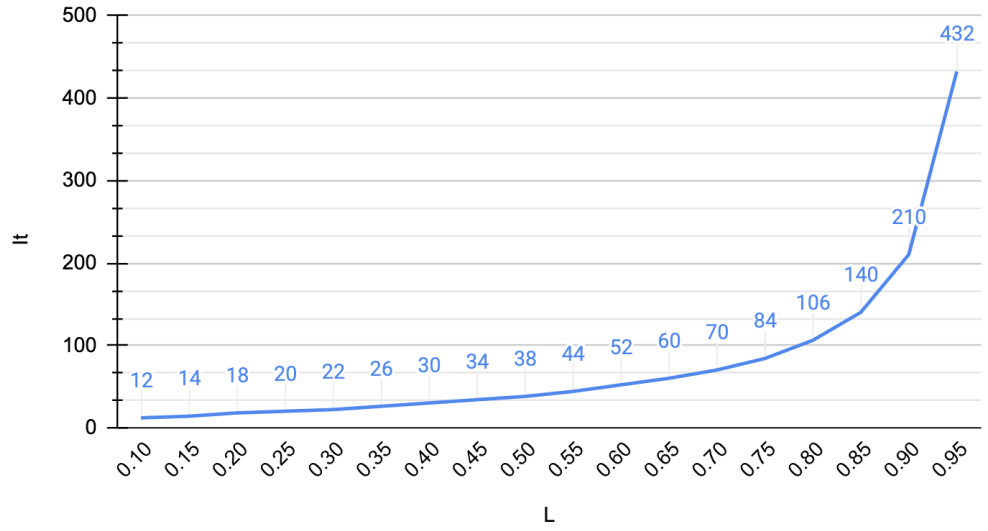

Figura 2: Gráfico iteraciones respecto el valor del damping factor

Como podemos apreciar en el gráfico, el número de iteraciones aumenta exponencialmente, llegando a 432 iteraciones con el valor más próximo a 1. Esto refleja como más beneficiamos la posibilidad de saltar de manera aleatoria de un nodo a otro hace que tardemos menos en converger y encontrar el *PageRank*. Cuando es cercano a 1, la probabilidad de seguir un enlace saliente es alta, lo que significa que el proceso de navegación tiende a quedarse en una región específica del grafo.

Experimentació tiempo

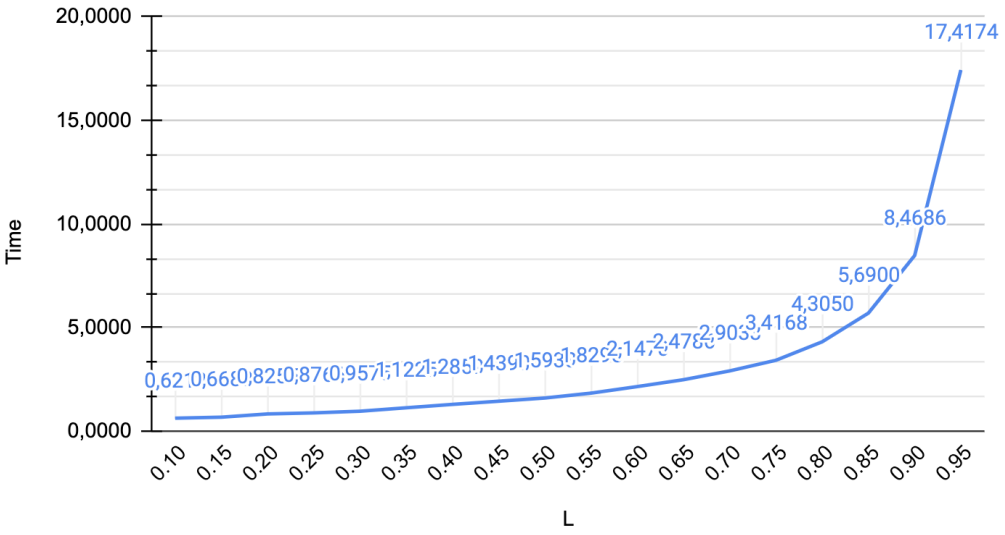

Figura 3: Gráfico tiempo, en segundos, respecto el valor del damping factor

Primero vemos que la curva dibujada por la Figura 1 sigue la tendencia de la Figura 2. Vemos la exponencialidad de los valores llegando a los 17 segundos al tener un valor cercano a 1, por las razones explicadas anteriormente.

Experimentació remaining *PageRank*

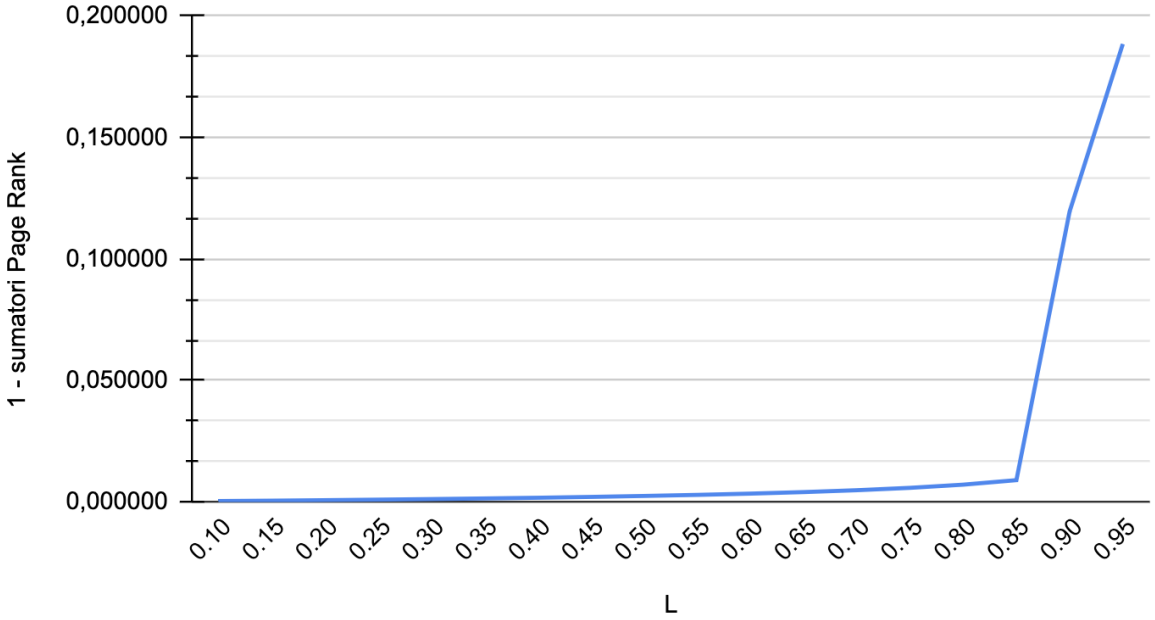

Figura 4: Gráfico remaining PageRank, respecto el valor del damping factor

Por último tenemos el PageRank que falta por repartir entre los nodos. Vemos que el aumento es mínimo hasta llegar a un dumping factor de 0.85, a partir de allí vemos que se dispara.

Conclusiones

Vemos después de estudiar los diferentes valores del dumping factor, que el valor que se nos da de manera predefinida, 0.85, es el valor más óptimo en cuanto a iteraciones, tiempo y distribución de *PageRank*. Es importante tener en cuenta que es necesario tener un dumping factor algo elevado, con un L más alto, el algoritmo pone más énfasis en la estructura del grafo y en cómo los nodos están conectados entre sí, sino nos estamos basando solo en la aleatoriedad. Al fin i el cabo, queremos reflejar una preferencia por seguir los enlaces salientes.