Álgebra Lineal Computacional

1er Cuatrimestre 2024

Trabajo Práctico N° 1: El problema de PageRank.

Introducción

El motor del buscador de Google, desde hace más de 20 años, utiliza el denominado *ranking de Page* o *PageRank*[1] como uno de los criterios para ponderar la importancia de los resultados de cada búsqueda. Calcular este ranking requiere "simplemente" resolver un sistema de ecuaciones lineales donde la cantidad de ecuaciones e incógnitas del sistema es igual al número de páginas consideradas.

Modelado del problema

Para un determinado conjunto de n páginas web se define la *matriz de conectividad* $\boldsymbol{W}$ estableciendo $w_{ij} = 1$ si la página j tiene un link a la página i y $w_{ij} = 0$ en caso contrario. Además $w_{ii} = 0$ pues ignoramos los *autolinks*. De esta forma, la matriz $\boldsymbol{W}$ puede resultar extremadamente rala y muy grande de acuerdo al tamaño del conjunto.

Para cada página $j = 1, \ldots, n$, definimos su *grado* como:

$$c_j = \sum_{i=1}^{n} w_{ij} \tag{1}$$

Es decir, la cantidad de links *salientes* de j. Típicamente c_j es un número mucho menor que n.

Se busca que el ranking sea mayor en las páginas *importantes*. Heurísticamente, una página es importante cuando recibe muchos "votos" de otras páginas, es decir, links. Pero no todos los links pesan igual: los links de páginas más importantes valen más. Pocos links de páginas importantes pueden valer más que muchos links de páginas poco importantes. Y los links de páginas con muchos links valen poco. Por lo tanto, una forma de calcular la importancia o *puntaje* x_i de la página i es:

$$x_i = \sum_{j=1}^{n} \frac{x_j}{c_j} w_{ij} \tag{2}$$

Es decir, la página j le aporta a i su puntaje ponderado por cuántos links salientes tiene. También, se puede definir la matriz de puntajes $\boldsymbol{R} = \boldsymbol{W}\boldsymbol{D}$, donde $\boldsymbol{D}$ es una matriz diagonal con elementos d_{jj} de la forma:

$$d_{jj} = \begin{cases} 1/c_j & \text{si} \quad c_j \neq 0 \\ 0 & \text{si} \quad c_j = 0 \end{cases},$$

[1] Por Larry Page, uno de los fundadores de Google, otrora joven científico actualmente devenido multimillonario. Ver artículo original del 1998 con más de 16500 citas [1].

Lo cual nos permite calcular el ranking de todas las páginas como:

$$\boldsymbol{R}\ \boldsymbol{x} = \boldsymbol{x} \tag{3}$$

donde $\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_n)^t$. Luego, la ecuación 2 corresponde al elemento i-ésimo $(\boldsymbol{R}\,\boldsymbol{x})_i$.

Este modelo tiene un problema: no logra capturar el comportamiento errático del usuario mientras navega por la red redes.

Modelo del navegante aleatorio

Un enfoque alternativo es considerar el modelo del *navegante aleatorio*. El navegante aleatorio empieza en una página cualquiera del conjunto, y luego en cada página j que visita elige con probabilidad $p \in (0, 1)$ si va a seguir uno de sus links, o con probabilidad $1 - p$, si va a pasar a otra página cualquiera del conjunto. Una vez tomada esa decisión, si decidió seguir un link de la página j elige uno al azar con probabilidad $1/c_j$, mientras que si decidió pasar a otra página cualquiera entonces elige una al azar con probabilidad $1/n$. Cuando la página j no tiene links salientes, es decir $c_j = 0$, elige al azar una página cualquiera del conjunto. Por lo tanto, se espera que luego de mucho surfear el navegante aleatorio va a estar en páginas importantes con mayor probabilidad.

Formalmente, definimos la matriz $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{n \times n}$ donde a_{ij} representa la probabilidad de pasar de la página j a la página i:

$$a_{ij} = \begin{cases} (1-p)/n + (p\ w_{ij})/c_j & \text{si} \quad c_j \neq 0 \\ 1/n & \text{si} \quad c_j = 0 \end{cases}, \tag{4}$$

Entonces el *ranking de Page* es la solución del sistema:

$$\boldsymbol{A}\ \boldsymbol{x} = \boldsymbol{x} \tag{5}$$

que cumple $x_i \geq 0$ y $\sum_i x_i = 1$.

Por lo tanto, el elemento i-ésimo $(\boldsymbol{A}\boldsymbol{x})_i$ es la probabilidad de encontrar al navegante aleatorio en la página i sabiendo que x_j es la probabilidad de encontrarlo en la página j, para $j = 1, \ldots, n$.

Luego, la matriz $\boldsymbol{A}$ puede reescribirse como:

$$\boldsymbol{A} = p\ \boldsymbol{W}\ \boldsymbol{D} + \boldsymbol{e}\ \boldsymbol{z}^t,$$

donde $\boldsymbol{D}$ es una matriz diagonal de la forma

$$d_{jj} = \begin{cases} 1/c_j & \text{si} \quad c_j \neq 0 \\ 0 & \text{si} \quad c_j = 0 \end{cases},$$

$\boldsymbol{e}$ es un vector columna de unos de dimensión n y $\boldsymbol{z}$ es un vector columna cuyos componentes son:

$$z_j = \begin{cases} (1-p)/n & \text{si} \quad c_j \neq 0 \\ 1/n & \text{si} \quad c_j = 0 \end{cases}.$$

Así, la ecuación (5) puede reescribirse como

$$(\boldsymbol{I} - p\ \boldsymbol{W}\ \boldsymbol{D})\ \boldsymbol{x} = \gamma\ \boldsymbol{e}, \tag{6}$$

donde $\gamma = \boldsymbol{z}^t\ \boldsymbol{x}$ funciona como un factor de escala.

Cálculo del ranking de Page

De esta manera, un procedimiento para calcular el ranking de Page consiste en:

1. Suponer $\gamma = 1$.
2. Resolver el sistema lineal de la ecuación (6).
3. Normalizar el vector $\boldsymbol{x}$ de manera que $\sum_i x_i = 1$.

Enunciado

Se debe implementar un programa en `Python` que realice el cálculo del ranking de Page según el procedimiento descripto anteriormente.

Como parte **obligatoria** se pide implementar la factorización LU para resolver el sistema de ecuaciones (6) que permite hallar la solución buscada (es decir, el ranking de páginas). Se podrá utilizar la función `solve_triangular` de la librería `SciPy` de `Python` para resolver sistemas triangulares.

Previamente, **deberán** estudiar las características de la matriz involucrada y responder a lo siguiente:

1. ¿Por qué la matriz $\boldsymbol{A}$ definida en (4) es equivalente a $p\, \boldsymbol{W}\, \boldsymbol{D} + \boldsymbol{e}\, \boldsymbol{z}^t$? Justificar adecuadamente.
2. ¿Cómo se garantiza existencia de la factorización LU? ¿ La matriz $(\boldsymbol{I} - p\, \boldsymbol{W}\, \boldsymbol{D})$ está bien condicionada? ¿Cómo influye el valor de p?

Experimentación

Se deberá realizar tanto un análisis cualitativo como cuantitativo de los métodos vistos en el trabajo.

1. Para el análisis cuantitativo, se pide, como mínimo, estudiar los tiempos de procesamiento en función del tamaño del grafo de páginas y de la densidad del mismo. Para esto, se espera que presenten gráficos mostrando los tiempos de ejecución para obtener la solución en función de la cantidad de nodos/links de diferentes grafos de páginas aleatorios.
2. Para el análisis cualitativo se deberán estudiar los rankings obtenidos, en función de la estructura del grafo, y del valor de p. Para esto, se espera que presenten gráficos mostrando las probabilidades de las páginas mejor rankeadas en función del valor de p.
3. Se considerarán los siguientes casos de prueba:
   - `instagram_famosos_grafo.txt`,
   - `mathworld_grafo.txt`,
   - `test_dosestrellas.txt`,
   - `test_15_segundos.txt`.
   - `test_30_segundos.txt`.

   Para el caso `test_dosestrellas.txt` se pregunta: ¿Cuál es la mínima cantidad de links que se deben agregar para que la pagina 1 quede primera en el ranking? ¿Cómo se modificó la conectividad? Analizar.

El grupo **deberá** proponer al menos 3 instancias de prueba, dos de las cuales deben ser de tipo TODOS CONECTADOS, y otra de tipo NINGUNO CONECTADO. La tercera instancia queda a criterio del grupo.

Para el análisis, guiarse y responder las siguientes preguntas:

1. ¿Cómo es el ranking obtenido en cada caso de acuerdo a la estructura del grafo páginas?
2. ¿Qué conclusiones pueden sacar de la interpretación de los resultados?

Los archivos de entrada serán de texto plano, y respetan el siguiente formato: en la primera línea un entero N, la cantidad total de páginas. En la segunda línea un entero M, la cantidad total de links. Luego siguen M líneas, cada una con dos enteros i j separados por un espacio ($1 \leq i, j \leq N$), indicando que hay un link de la página i a la página j.

Entrega y lineamientos

La entrega se realizará a través del campus virtual de la materia con las siguientes fechas y formato:

- Fecha de entrega: hasta el miércoles **8 de mayo** a las 23:59 hs.
- Fecha de re-entrega: hasta el **28 de mayo** a las 23:59 hs.
  Es condición obligatoria haber realizado un envío en la primera fecha de entrega para tener la posibilidad de reentregar.
- Formato: Jupyter Notebook del template-alumnos. Archivo funciones.py completo.

Prestar especial atención a las siguientes indicaciones:

- El TP1 se realizará en grupos de dos personas. Deberán inscribir su grupo en el foro 'Foro de Grupos de TP' destinado para tal fin, dentro de la sección Laboratorio/TP1 del campus de la materia.

  Importante: es indispensable realizar la inscripción previa del grupo para poder hacer el envío a través del campus. Los grupos o personas no inscriptas en grupos no estarán habilitadas en el formulario de carga del TP. No se aceptarán envíos por email.

- Leer el enunciado completo antes de comenzar a generar código y sacarse todas las dudas de cada ítem antes de implementar. Para obtener un código más legible y organizado, pensar de antemano qué funciones deberán implementarse y cuáles podrían reutilizarse.

- El código debe estar correctamente comentado. Cada función definida debe contener un encabezado donde se explique los parámetros que recibe y qué se espera que retorne. Además las secciones de código dentro de la función deben estar debidamente comentados. Los nombre de las variables deben ser explicativos.

- Las conclusiones y razonamientos que respondan los ejercicios, o cualquier experimentación agregada, debe estar debidamente explicada en bloques de texto de las notebooks (markdown cells), separado de los bloques de código. Aprovechen a utilizar código LaTeX si necesitan incluir fórmulas.

- Gráficos: deben contener título, etiquetas en cada eje y leyendas indicando qué es lo que se muestra.

Referencias

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.