DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

Ranking de páginas web y equipos deportivos

Métodos Numéricos

Segundo Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Alvarez, Lautaro Leonel	268/14	lautarolalvarez@gmail.com
Maddonni, Axel Ezequiel	200/14	axel.maddonni@gmail.com
Thibeault, Gabriel Eric	114/13	gabriel.eric.thibeault@gmail.com

En este trabajo estudiaremos diversos métodos de ranking de páginas web, prestando particular atención al algoritmo Page Rank de Brin y Page. También realizaremos un estudio similar en el contexto de equipos deportivos.

Page Rank, Método de la Potencia

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://dc.uba.ar

Índice

1. Introducción Teórica

En este trabajo estudiaremos algoritmos que buscan ordenar páginas web de acuerdo a su importancia relativa dentro de la red. Nos concentraremos en el algoritmo de PageRank, que constituye uno de los criterios utilizados para ponderar la importancia de los resultados de una búsqueda, y distinguió al motor de búsqueda de Google en su lanzamiento, debido a la calidad de sus resultados. Además, compararemos los resultados obtenidos por otros algoritmos más simples como IN-DEG, resaltando la diferencia en tiempos de ejecución, calidad de resultados, entre otras. En particular, veremos las vulnerabilidades de In-Deg frente a ciertos ataques posibles, y su falla a la hora de medir la importancia relativa de las páginas comparado con PageRank.

Por otra parte, estudiaremos la aplicación de estos algoritmos usados en un contexto muy diferente, como lo son las competencias deportivas. Éstas requieren inevitablemente la confección de Rankings o Tablas de Posiciones, basados en determinadas reglas según cada deporte. Para ello, veremos una manera de modelar estas competencias de modo que al aplicar PageRank logremos resultados similares a los que arrojaría el verdadero ranking de posiciones, y en qué casos vemos diferencias, en particular para la competencia de primera división de fútbol argentino de la AFA.

1.1. Page Rank y Páginas Web

El algoritmo PageRank se basa en la construcción del siguiente modelo. Supongamos que tenemos una red con n páginas web $Web = \{1, \ldots, n\}$ donde el objetivo es asignar a cada una de ellas un puntaje que determine la importancia relativa de la misma respecto de las demás. Para modelar las relaciones entre ellas, definimos la *matriz de conectividad* $W \in \{0,1\}^{n\times n}$ de forma tal que $w_{ij} = 1$ si la página j tiene un link a la página i, y $w_{ij} = 0$ en caso contrario. Además, ignoramos los *autolinks*, es decir, links de una página a sí misma, definiendo $w_{ii} = 0$. Tomando esta matriz, definimos el grado de la página i, $deg(i)$, como la cantidad de links salientes hacia otras páginas de la red. Además, notamos con x_i al puntaje asignado a la página $i \in Web$, que es lo que buscamos calcular.

Consideraremos que la importancia de la página v obtenida mediante el link de la página u es proporcional a la importancia de la página u e inversamente proporcional al grado de u. Si la página u contiene $deg(u)$ links, uno de los cuales apunta a la página v, entonces el aporte de ese link a la página v será $x_u/deg(u)$. Luego, sea $L_k \subseteq Web$ el conjunto de páginas que tienen un link a la página k. Para cada página pedimos que

$$x_k = \sum_{i\in L_k} \frac{x_i}{deg(i)}, \qquad k = 1, \ldots, n. \tag{1}$$

Una formulación equivalente es pensar en términos de un *navegante aleatorio* en la web. Si consideramos al navegante visitando la página i en un instante de tiempo, en el próximo paso, el navegante elige entre los links disponibles con probabilidad $1/deg(i)$. El ranking de la página i está definido por la probabilidad de que en un instante particular $k > K$, el navegante está en la página i. Para un K suficientemente grande, y con algunas modificaciones a la navegación aleatoria, ésta probabilidad es única. Considerando la cadena de markov inducida por la navegación aleatoria en la Web, la matriz que describe la transición de j a i está dada por P con $p_{ij} = 1/deg(j)$ si $w_{ij} = 1$, y $p_{ij} = 0$ en caso contrario. Luego, el modelo planteado en (1) es equivalente a encontrar un $x \in \mathbb{R}^n$ tal que $Px = x$, es decir, encontrar (suponiendo que existe) un autovector asociado al autovalor 1 de una matriz cuadrada, tal que $x_i \geq 0$ y $\sum_{i=1}^{n} x_i = 1$.

Para que P sea una matriz de probabilidades de transición válida, todos las páginas deben tener al menos un link de salida, es decir, P no debe tener columnas de todos ceros. En ese caso, consideramos que si la página no tiene links de salida, el navegante aleatorio pasa a cualquiera de las páginas de la red con probabilidad $1/n$. Para representar esta situación, definimos $v \in \mathbb{R}^n$, con $v_i = 1/n$ y $d \in \{0,1\}^n$ donde $d_i = 1$ si $deg(i) = 0$, y $d_i = 0$ en caso contrario. La nueva matriz de transición es

$$\begin{aligned} D &= vd^t \\ P' &= P + D. \end{aligned}$$

Sin embargo, para nuestro algoritmo, es deseable que los rankings resultantes sean únicos. Para eso, debe existir un único autovector $x \in \mathbb{R}^n$ tal que $P'x = x$, con x vector de probabilidad que usaremos

para determinar la importancia de cada página. Esto sucede si la matriz de transición está *fuertemente conectada*, es decir, la matriz además de ser estocástica por columnas es positiva. La demostración se puede encontrar en el paper de Bryan y Leise [4], sección 3.2. Una interpretación más intuitiva de esto se puede ver en el caso con una red dividida en dos o más subredes, donde un navegante en una de las subredes no puede llegar a través de los links a las páginas de la otra red. En este caso, no sería posible comparar la importancia entre las páginas de las distintas subredes, por lo que es probable que existan más de un vector solución.

La forma estándar para asegurar esta propiedad es agregar un set de transiciones con poca probabilidad a todos los nodos. En términos del navegante aleatorio, que dado que se encuentra en una página cualquiera, este pueda visitar otra página del conjunto con poca probabilidad, independientemente de si esta se encuentra o no entre los links salientes de la actual (fenómeno conocido como *teletransportación*). Para ello consideramos que esta decisión se toma con probabilidad $c \geq 0$, y podemos incluirlo al modelo de la siguiente forma:

$$\begin{aligned} E &= v\bar{1}^t \\ P'' &= cP' + (1-c)E, \end{aligned}$$

donde $\bar{1} \in \mathbb{R}^n$ es un vector tal que todas sus componentes valen 1.

A partir de aquí nos referiremos a la matriz P'' simplemente como la matriz A. Probabilísticamente, la componente x_j del vector solución (normalizado) del sistema $Ax = x$ representa la proporción del tiempo que, en el largo plazo, el navegante aleatorio pasa en la página $j \in Web$. Denotaremos con π al vector solución de la ecuación $Ax = x$, que es comúnmente denominado *estado estacionario*, y es exactamente el vector que se busca computar.

El algoritmo de PageRank computa el autovector principal empezando con la distribución uniforme y calculando sucesivas iteraciones $x^k = A\ x^{k-1}$ hasta converger. (Numéricamente, hasta que la diferencia de los x que vamos obteniendo sea menor a un determinado valor epsilon pasado por parámetro.). Este método es conocido como Método de la Potencia.

1.1.1. Método de la Potencia

El método de la potencia es el método más viejo para computar el autovector principal de una matriz. La convergencia del método se puede ver de la siguiente manera intuitivamente:

Por simplicidad, asumimos que el vector inicial x^0 pertenece al subespacio formado por los autovectores de A, y por ende puede escribirse como una combinación lineal de autovectores de A:

$$x^0 = u_1 + \alpha_2 u_2 + ... + \alpha_m u_m$$

Como sabemos que el primer autovalor de una Matriz de Markov es $\lambda_1 = 1$, tenemos que:

$$x^1 = Ax^0 = u_1 + \alpha_2\lambda_2 u_2 + ... + \alpha_m\lambda_m u_m$$

y

$$x^n = A^n x^0 = u_1 + \alpha_2\lambda_2^n u_2 + ... + \alpha_m\lambda_m^n u_m$$

Como $\lambda_n \leq ... \leq \lambda_2 < 1$, $A^n x^0$ se acerca a u_1 mientras n crece. De esta manera, el método de la potencia converge al autovector principal de matriz de Markov A.

Notar que si λ_2 es un valor cercano a 1, el método converge más lentamente, porque n debe ser suficientemente grande para que λ_2^n converja a 0, y viceversa.

Una iteración del método de la potencia consiste en una multiplicación entre matriz y vector Ax^k. Generalmente, esto costa $O(n^2)$ operaciones. Sin embargo, en el paper de Kamvar presentan un algoritmo para aprovechar la matriz esparsa P para calcular las multiplicaciones en costo $O(n)$.

El algoritmo propuesto por Kamvar [6] para calcular $y = Ax$ es el siguiente:

$y = cPx$
$w = ||x||_1 - ||y||_1$
$y = y + w.v$

donde v es el vector uniforme de dimensión n: $[1/n]_{nx1}$.

1.1.2. Page Rank con personalización

Este método consiste en una variante del algoritmo detallado en la sección anterior. Page Rank le asigna la misma probabilidad a cada link que sale de una misma página, y la misma probabilidad de teletransportarse a todas las páginas. La idea de esta variante es aprovechar el historial de cada usuario, ponderando los links a páginas que ya visitó y asignándole una mayor probabilidad de teletransportación a tales sitios. El peso que se le da a la personalización está dado por un parámetro, que denominaremos *"p"*, análogo de personalización al parámetro *"c"* de teletransportación.

1.1.3. Demostración de la correctitud del Algoritmo de Kamvar

Primero desarrollamos la fórmula de la y buscada, que resuelve el sistema, utilizando las fórmulas de las matrices introducidas previamente:

$$\begin{aligned} y &= Ax \\ &= (P'')x = (cP' + (1-c)E)x \\ &= cP'x + (1-c)Ex \\ &= c(P+D)x + (1-c)Ex \\ &= cPx + cDx + (1-c)Ex \end{aligned}$$

Queremos ver que $cDx + (1-c)Ex = wv$. El producto de matrices queda:

$$\begin{aligned} cDx + (1-c)Ex &= c \begin{bmatrix} v_1d_1 & v_1d_2 & ... & v_1d_n \\ v_2d_1 & v_2d_2 & ... & v_2d_n \\ ... & ... & ... & ... \\ v_nd_1 & v_nd_2 & ... & v_nd_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} + (1-c) \begin{bmatrix} v_1 & v_1 & ... & v_1 \\ v_2 & v_2 & ... & v_2 \\ ... & ... & ... & ... \\ v_n & v_n & ... & v_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix} \\ &= c\sum_{j=1}^{n}(v_id_jx_j) + (1-c)\sum_{j=1}^{n}(v_ix_j) \quad \forall i(para\ cada\ fila\ i\ del\ vector\ resultante) \\ &= c\ v_i\sum_{j=1}^{n}(d_jx_j) + (1-c)v_i\sum_{j=1}^{n}(x_j) \\ &= v_i\ (c\sum_{j=1}^{n}(d_jx_j) + (1-c)\sum_{j=1}^{n}(x_j)) \\ &= v_i\ (\sum_{j=1}^{n}(x_j) + c\ (\sum_{j=1}^{n}(d_jx_j) - \sum_{j=1}^{n}(x_j))\ ) \\ &= v_i\ (\sum_{j=1}^{n}(x_j) + c\sum_{j=1}^{n}(d_jx_j - x_j)\ ) \\ &= v_i\ (\sum_{j=1}^{n}(x_j) + c\sum_{j=1}^{n}(x_j(d_j-1))\ ) \\ &= v_i\ (\sum_{j=1}^{n}(x_j) - c\sum_{j=1}^{n}(x_j(1-d_j))\ ) \\ &= v_i\ (\sum_{j=1}^{n}(x_j) - c\sum_{\substack{j=1 \\ d_j=0}}^{n}(x_j)\ ) \end{aligned}$$

Desarrollamos wv, también de dimensión n, y vemos que llegamos a lo mismo:

$$\begin{aligned} wv &= v_i(||x||_1 - ||y||_1) \ \ \forall i \ \ (para\ cada\ fila\ i\ del\ vector\ resultante) \\ &= v_i(||x||_1 - ||cPx||_1) \\ &= v_i(||x||_1 - c|||Px||_1) \ \ (c\ es\ positivo) \\ &= v_i(\sum_{j=1}^{n} x_j - c\ (\sum_{i=1}^{n} \sum_{\substack{j=1 \\ w_{ij}=1}}^{n} \frac{x_j}{deg(j)})\ )\ \ (no\ escribo\ los\ módulos\ pues\ son\ valores\ positivos) \end{aligned}$$

Desarrollando las sumatorias, se puede observar que la norma 1 del vector resultante de multiplicar la matriz P por el vector x se puede reescribir sacando factor común x_i, multiplicados por la sumatoria de todos los $\frac{1}{deg(i)}$ tales que $w_{ij} = 1$. Esta sumatoria suma 1 en caso de que la página i tenga links salientes ($d_i = 0$), pues son las probabilidades sumadas de saltar desde una página i a sus $deg(i)$ páginas apuntadas. En caso de que la página i no tenga links salientes, es decir $d_i = 1$, como la columna i-ésima de la matriz P es de todos ceros en la multiplicación se anula el x_i.

$$\begin{aligned} &= v_i(\sum_{j=1}^{n} x_j - c\ (\sum_{i=1}^{n} x_i \sum_{\substack{j=1 \\ w_{ij}=1}}^{n} \frac{1}{deg(i)})\ ) \\ &= v_i\ (\sum_{j=1}^{n}(x_j) - c \sum_{\substack{i=1 \\ d_i=0}}^{n}(x_i)\ ) \\ &= cDx + (1-c)Ex \end{aligned}$$

1.2. Competencias Deportivas

1.2.1. Método GeM

El método GeM estudia eventos deportivos estableciendo buscando establecer un ranking de los competidores en base a los resultados obtenidos. Se basa en tomar la liga a estudiar como un grafo y a los competidores como nodos de ese grafo. Los links salientes de un equipo representan una derrota y tienen un peso determinado por la diferencia de goles del encuentro. En cierto modo se lo puede asimilar al Page Rank tomando a los equipos como páginas y los resultados de los encuentros como links. EL método consiste en una serie de pasos que vamos a explicar a continuación y concluye aplicando el método Page Rank antes visto.

Se comienza armando una matriz $A^t \in \mathbb{R}^{n\times n}$ (siendo n = #competidores) tal que:

$$A^t_{ji} = \begin{cases} w_{ji} & \text{si el competidor } i \text{ fue derrotado por el } j. \\ 0 & \text{en caso de empate o victoria.} \end{cases}$$

donde w_{ji} es la diferencia del marcador entre i y j. En caso de mas de una derrota las diferencias se acumulan.

Luego definimos la matriz $H^t_{ji} \in \mathbb{R}^{n\times n}$ como

$$H^t_{ji} = \begin{cases} A^t_{ji} / \sum_{k=1}^{n} A^t_{ki} & \text{si } A_{ji} \neq 0. \\ 0 & \text{en caso contrario.} \end{cases}$$

Ahora tomamos la H^t=P y aplicamos el método Page Rank sobre P. Los pasos a seguir son los mismos que se explicaron previamente.

2. Desarrollo

2.1. Page Rank para Páginas Web

2.1.1. Algoritmo PageRank vs IN-DEG

El algoritmo de PageRank presentado cumple con lo especificado en la introducción teórica. Utilizando el método de la potencia explicado anteriormente, dada la matriz P" calculada a partir del input, y un parámetro de teleportación c pasado por parámetro, computa el primer autovector principal de probabilidad que muestra la "proporción.[en] el tiempo que un navegante aleatorio pasaría en cada página. A partir de ese vector, confeccionamos el Ranking que es devuelto en un archivo de salida.

Usando el Algoritmo propuesto por Kamvar, demostrado en la sección anterior, aprovechamos la esparsidad de la matriz P. Para ello utilizamos una estructura llamada *Compressed Column Storage* para optimizar el tiempo de ejecución de nuestro algoritmo. La estructura que aprovecha esta característica se encuentra explicada en la siguiente sección.

Por otra parte, el algoritmo IN-DEG que también utilizaremos en nuestros experimentos para comparar computa un Ranking solamente tomando en cuenta los enlaces entrantes a una página, ordenándolas por importancia de manera decreciente.

2.1.2. Representación de Matrices Esparsas

En el enunciado de este trabajo se mencionaron tres estructuras distintas para aprovechar la esparsidad de la matriz P (detalla por Kamvar et al[6]). Éstas son: Dictionary of Keys (DoK), Compressed Column Storage (CCS) y Compressed Row Storage (CRS).

DoK consiste en guardar triplas de $< fila, columna, valor >$ para cada elemento no-nulo de la matriz. Permite construir la representación de la matriz incrementalmente y el acceso aleatorio rápido a los elementos de ésta (ya sea en tiempo $O(1)$ o $O(log(n))$ - con n igual a la cantidad de elementos no nulos-, dependiendo de la implementación elegida para el diccionario). Sin embargo, no es particularmente buena para realizar operaciones de matrices, y el overhead incurrido por esta estructura es mayor que el de las otras.

CCS y CRS mantienen tres arreglos: uno para los valores no-nulos (ordenados por columna y luego por fila en el caso de CCS y al revés en el caso de CRS), uno que indica a qué columna (en el caso de CCS, o fila para CRS) corresponden esos valores y uno que determina cúantos elementos de cada fila (para CCS, o columna para CRS) son no-nulos. CCS es eficiente (itera sólo sobre los elementos no-nulos, y el overhead es $O(1)$) para multiplicar por derecha (es decir, con la matriz CCS siendo la derecha), y CRS para multiplicar por izquierda.

La matriz P se puede construir a partir de la entrada en una CRS, o se puede construir P^T como CCS (la representación de una matriz A cualquiera en CRS es idéntica a la representación CCS de A^T). Nosotros necesitamos P^T como CRS, por lo que deberemos pasar la representación CCS de P^T a CRS. Afortunadamente, esto se puede realizar en tiempo lineal en la cantidad de elementos no-nulos (equivalente al tiempo necesario para transponer la matriz, por lo que no se incurre en overhead, desde un punto de vista asintótico).

2.2. Problema del Page Rank para Competencias Deportivas

2.2.1. Algoritmo GeM

Para implementar el algoritmo GeM no tuvimos grandes problemas. Decidimos hacer una implementación paralela al Page Rank para evitar complicaciones, ya que sabemos que contamos con las siguientes características:

- Una cantidad de datos acotados: la matriz está definida por la cantidad de equipos, pero sabemos que es acotada y que no nos va a generar grandes problemas.
- En la liga que vamos a analizar los equipos se enfrentar todos contra todos, por lo que no vamos a toparnos con una matriz esparsa. Por eso nos vamos a ahorrar las optimizaciones de estructuras (mas aún contando con el item anterior).

- Los resultados son números enteros y de rango acotado (acotados por el juego en sí, ya que sería un caso borde que un equipo convierta mas de 6 goles en un partido).

Analizando los puntos nombrados decidimos tomar un vector de vectores de double (matriz de double "M"). Vamos recorriendo el archivo de entrada donde se encuentran los resultados de los distintos partidos y aplicamos la siguiente lógica:

1: **if** goles(j) == goles(i) **then**
2: //salteamos el partido
3: **else if** goles(j) >goles(i) **then**
4: $M_{ij} = M_{ij} + (goles(j) - goles(i))$
5: **else**
6: $M_{ji} = M_{ji} + (goles(i) - goles(j))$
7: **end if**

Con este paso obtenemos la "matriz A^t". Luego la recorremos y dividimos a cada posición ij por la suma de todas las posiciones de su misma fila i. así obtenemos la "matriz H^t. Luego tomamos a P=H^t y realizamos los pasos del método Page Rank. Al finalizar imprimimos los resultados en el archivo de salida.

2.2.2. Método Estándar

El método estándar para la obtención del ranking de una liga de fútbol consta sumar los puntajes de los equipos en cada partido (3pts por ganar y 1pt por empatar). Para llevar esto a cabo tomamos un vector de enteros y recorremos el archivo de entrada aplicando el siguiente algoritmo para cada partido:

1: **if** goles(j) == goles(i) **then**
2: $M_{ij} = M_{ij} + 1$
3: $M_{ij} = M_{ji} + 1$
4: **else if** goles(j) >goles(i) **then**
5: $M_{ij} = M_{ij} + 3$
6: **else**
7: $M_{ji} = M_{ji} + 3$
8: **end if**

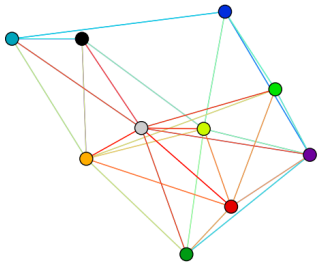
Al finalizar el proceso obtenemos el vector con los puntajes de cada equipo. Lo siguiente que hacemos es ordenarlo e imprimir los resultados en el archivo de salida.

3. Resultados y Discusión: PageRank

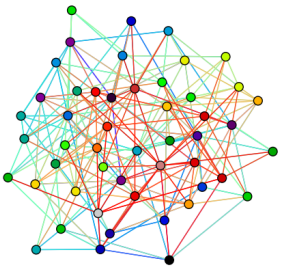
3.1. Experimentos sobre la convergencia de PageRank

3.1.1. Experimento 1

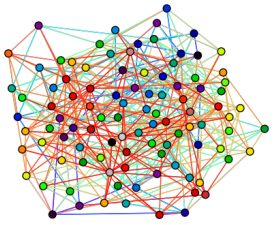
El primer experimento consiste en aumentar la cantidad de páginas del grafo, manteniendo constante la cantidad (en particular, 4) de links que salen de cada una de éstas. En las figuras 1(a), 1(b), 1(c), y 1(d) se pueden ver graficados algunos de los inputs. Cabe destacar que los links están relativamente distribuidos a lo largo del grafo: todas las páginas tienen la misma cantidad de links salientes, y si bien la cantidad de links entrantes puede variar (ya que esto fue generado aleatoriamente en el input), estadísticamente esta magnitud debería ser homogénea. Los colores no poseen ningún significado y sólo son incluidos por claridad (se vuelve díficil comprender los grafos monocromos).



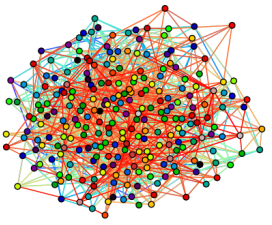
(a) Grafo del experimento 1, para un tamaño de 10 páginas



(b) Grafo del experimento 1, para un tamaño de 50 páginas



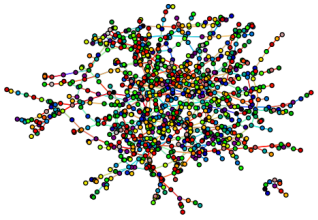
(c) Grafo del experimento 1, para un tamaño de 100 páginas



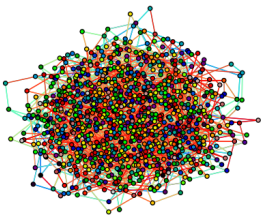
(d) Grafo del experimento 1, para un tamaño de 250 páginas

3.1.2. Experimento 2

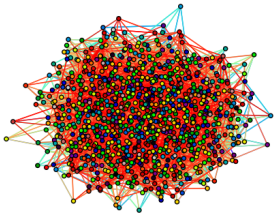
El experimento 2 es similar al primero: se observa cómo varía el tiempo de ejecución al variar la cantidad de elementos no-nulos de la matriz de Markov. La diferencia reside en modificar la cantidad de links que salen de cada página, manteniendo constante la cantidad total de páginas -en particular, 1000 páginas- (es decir, al revés que en el experimento 1). Si bien ambos experimentos miden lo mismo, el primero lo hace manteniendo de cierta forma la esparsidad de la matriz, mientras que el segundo lo hace modificándola. En las figuras 1(e), 1(f), y 1(g) se pueden ver graficados algunos de los inputs. Cabe destacar como el grafo se vuelve paulatinamente más interconectado.



(e) Grafo del experimento 2, para 1 link saliente por página



(f) Grafo del experimento 2, para 2 links salientes por página



(g) Grafo del experimento 2, para 4 links salientes por página

3.1.3. Resultados de los Experimentos 1 y 2

En las figuras 1, y 2 se pueden observar los resultados del experimento 1 (la segunda figura muestra sólo los primeros 4 valores, por claridad). En la figura 3, los del experimento 2. Para permitirnos mejor comparar los resultados de ambos experimentos, se han incluido los dos juntos en la figura 4 (la variable dependiente en este caso representa la cantidad de elementos no-nulos de la matriz).

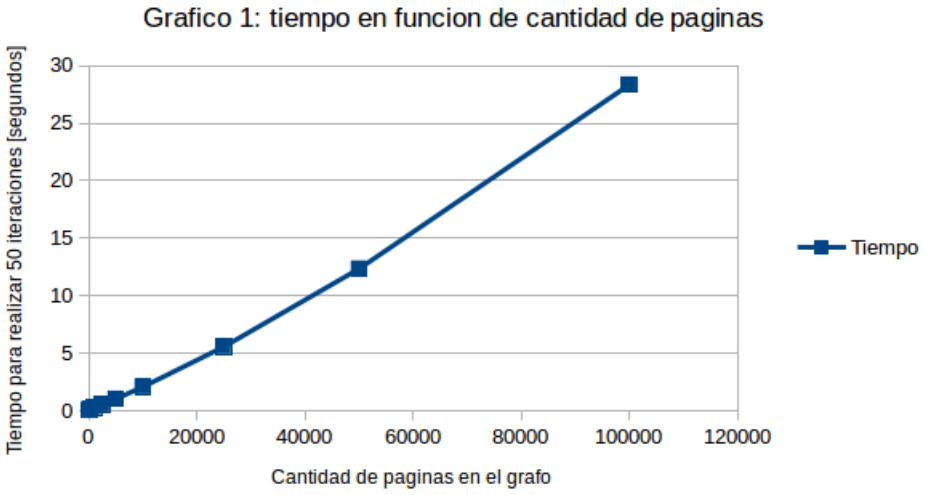


Figura 1: Resultado del experimento 1

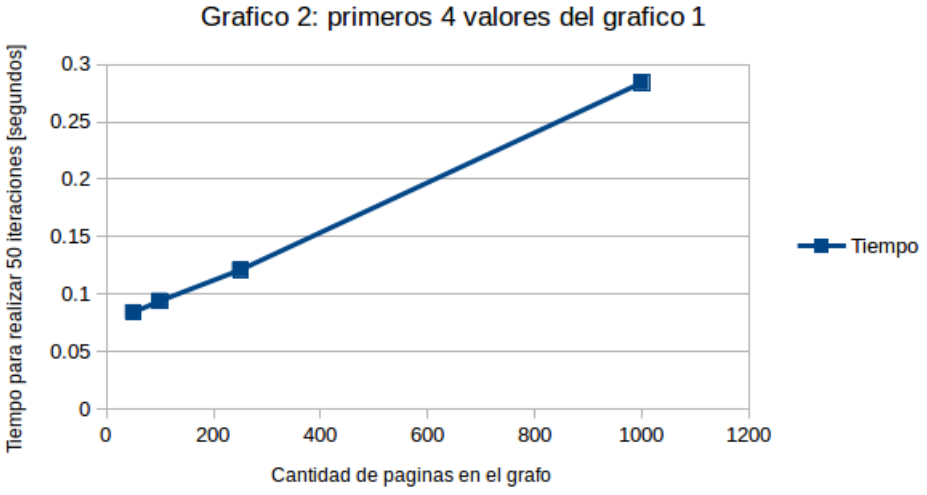


Figura 2: Resultado del experimento 1, primeros 4 valores

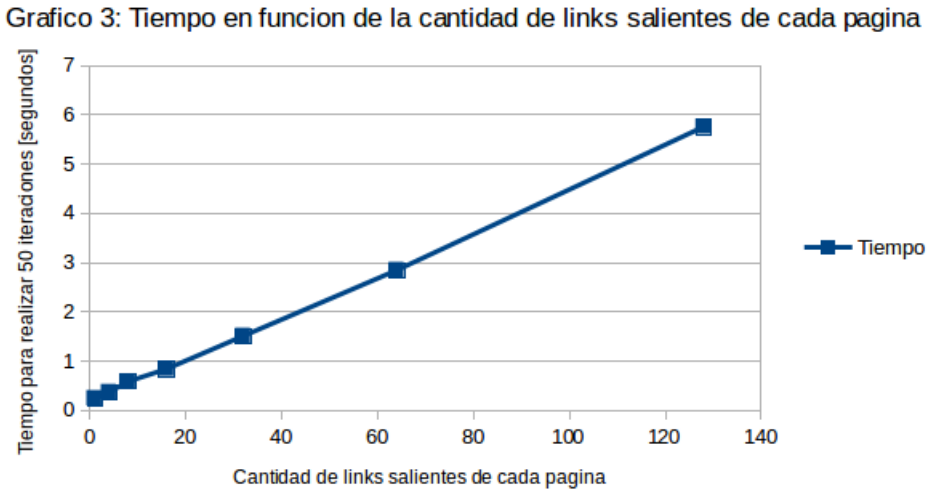


Figura 3: Resultado del experimento 2

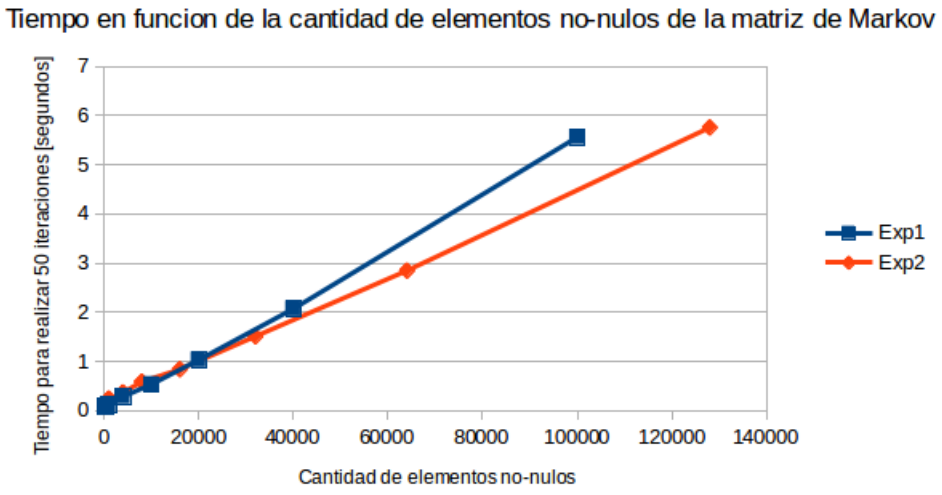


Figura 4: Resultado del experimento 1 y 2, juntos

Se puede ver que los resultados de ambos experimentos describen curvas similares, pero la pendiente del primero es más drástica. Esto es esperable, ya que si bien cada iteración del método de la potencia depende linealmente de la cantidad de elementos no-nulos de la matriz, también depende linealmente de la cantidad de páginas (el tamaño de los vectores uniformes utilizados está dado por la cantidad de nodos en el grafo, es decir de sitios en la red).

3.1.4. Experimento 3

Para el experimento 3, se medirá el tiempo de ejecución para una misma entrada (con 10000 páginas y 4 links salientes de cada una), variando el parámetro *"c"*.

Sabemos que *"c"* es igual al módulo del segundo autovalor de la matriz (como se encuentra demostrado en el paper *The Second Eigenvalue of the Google Matrix*[7] de Kamvar y Haveliwala). Como vimos en la introducción teórica, el método de la potencia trata obtener el primer autovalor al eliminar al resto en la combinación lineal, haciéndolos converger a 0 aplicando potenciación ya que son menores que 1. Cuanto mayor sea el c, y por ende, más cercano a 1 sea el módulo de λ_2, se espera que requiera de más iteraciones para que λ_2 converja a 0 (en términos del algoritmo, que sea menor que ε).

3.1.5. Experimento 4

En el experimento 4, mediremos el tiempo de ejecución para la una misma entrada (idéntica a la del tercer experimento) al variar el parámetro ε.

3.1.6. Resultados de los Experimentos 3 y 4

En las figuras 5 y 6 se encuentran los resultados de los experimentos 3 y 4, respectivamente. Cabe destacar que la escala utilizada para el parámetro ε en la figura 6 es logarítmica.

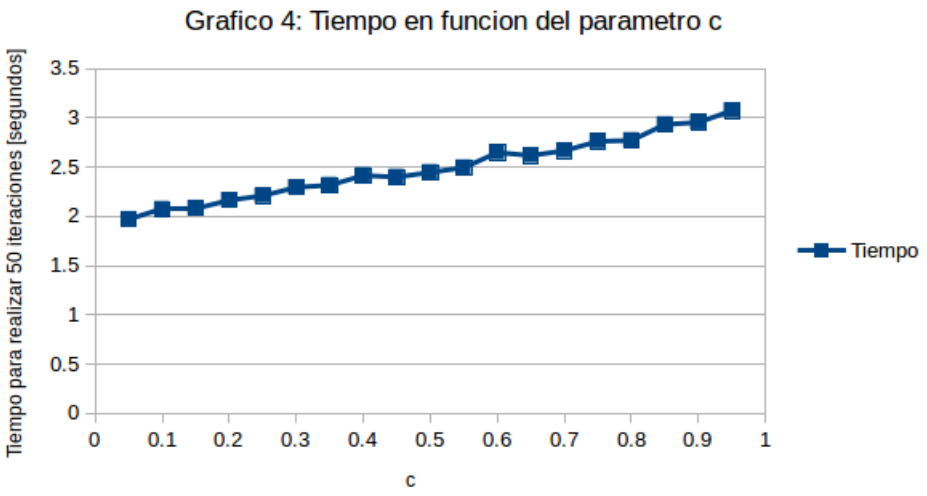


Figura 5: Resultado del experimento 3

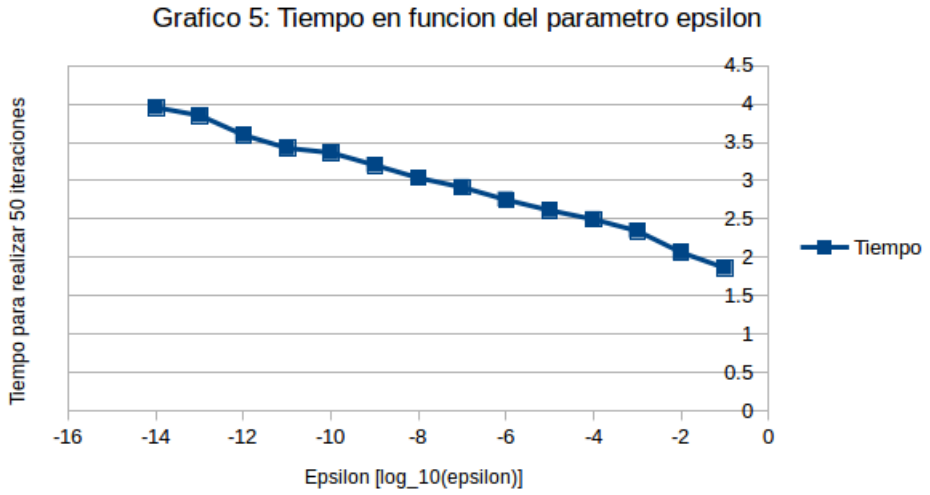


Figura 6: Resultado del experimento 4

Se ve claramente que al aumentar el parámetro *"c"* aumenta el tiempo de ejecución para computar el autovector principal. Lo mismo sucede al decrementar el ε, ya que disminuimos el margen que tomamos para traducir que los $|\lambda|$ convergen a 0.

Esto puede ser útil si se desea realizar PageRank con un *"c"* alto pero se debe calcular en un tiempo bajo, se puede incrementar ε (aunque se pierda un poco de presición).

Una observación es importante es que a pesar de que al disminuir el parámetro *"c"* el algoritmo se vuelve más rápido, aumenta la probabilidad de teletransportación y esto puede dar lugar a páginas con un alto ranking injusto. (ver Experimento 3 de la sección de resultados para Competencias Deportivas).

3.2. Experimentos sobre la eficacia de In-Deg, PageRank y PageRank con personalización

3.2.1. Experimento 5

El experimento 5 (y el 6) buscan comparar la eficiencia de PageRank e In-Deg en distintos escenarios. Son dos de los ejemplos que más comunmente se utilizan para detallar las ventajas de PageRank, y nuestra intención es comprobar que el algoritmo es efectivamente superior en estos casos.

En el caso del 5, la entrada consiste en un grafo donde hay una página "central" (como serían Google, Facebook, Wikipedia y demás) a la que apunta la mayoría de las páginas. El resto de las páginas se distribuyen uniformemente (es decir, mediante una variable aleatoria uniforme) los links salientes. La idea es que la página más importante debería ser la central, y las que la siguen en prioridad deberían ser páginas apuntadas por ésta. En la figura 7 se puede observar representado el input.

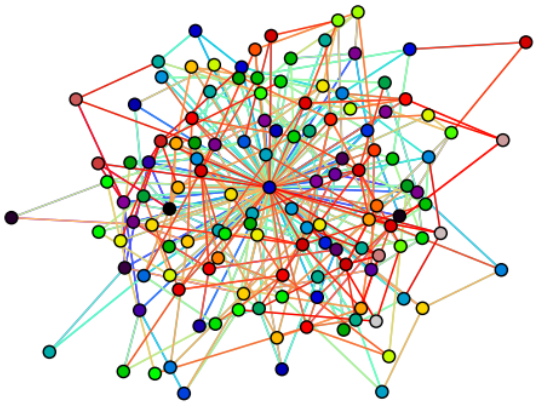


Figura 7: Grafo del experimento 5

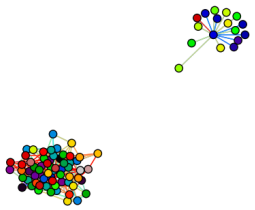
3.2.2. Resultados del Experimento 5

Tanto en In-Deg como en PageRank la página central es la primera en el ranking, pero en el caso de In-Deg la segunda es la página 36, que no es apuntada por la central. En cambio en PageRank todas las primeras páginas son apuntadas por la central, como es deseado.

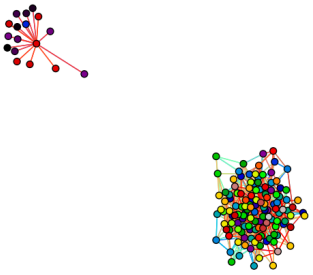
3.2.3. Experimento 6

El experimento 6 consiste en juzgar la susceptibilidad de los métodos a sucumbir bajo cierto tipo de ataque malicioso, donde una página crea otros sitios "títeres" que apuntan a ella para aumentar su ranking. Se espera que In-Deg sea más vulnerable que PageRank, ya que estas páginas maliciosas no son apuntadas por nadie (por lo que tienen un ranking inferior desde la perspectiva de PageRank, y por ende tendrán menor influencia en el orden final).

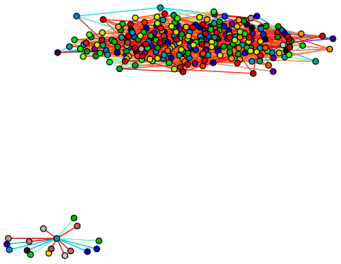
Se realizarán distintas iteraciones, manteniendo constante la cantidad de páginas atacantes (en particular, 16 títeres apuntando a un atacante), y variando la cantidad de páginas normales, para ver como cambian los resultados al modificar la proporción de páginas atacantes y normales. En las figuras 8(a), 8(b) y 8(c) se pueden observar los grafos que representan los distintos inputs.



(a) Grafo del experimento 6, para 64 páginas normales



(b) Grafo del experimento 6, para 128 páginas normales



(c) Grafo del experimento 6, para 256 páginas normales

3.2.4. Resultados del Experimento 6

En el caso de In-Deg, para las 3 entradas, la página atacante está en la primera posición del ranking. En cambio, para PageRank, se encuentra en las posiciones 39, 77 y 153 (de un total de 81, 145 y 273, respectivamente). Como se esperaba, PageRank es mucho menos vulnerable que In-Deg a este tipo de ataque.

3.2.5. Experimento 7

El séptimo experimento busca juzgar la efectividad de nuestra variante de personalización para el algoritmo PageRank. Como entrada se utilizará un grafo con 3 páginas centrales (que se puede observar en la figura 8. La idea es representar el resultado de un individuo buscando cierta expresión en nuestro motor de búsqueda (por ejemplo "Type of lambda expressions in C++11"), y obteniendo como resultado ciertas páginas, algunas de las cuales son "centrales" (como por ejemplo *cplusplus.com*, *cppreference.com* y *stackoverflow.com*) y algunas son de menor importancia (por ejemplo blogs y otros sitios de ese tipo).

PageRank le asignará un orden a estas páginas, seguramente con las tres centrales arriba del ranking, pero cada usuario tendrá su propia preferencia de sitios. Idealmente, el orden de los resultados de cada usuario tendrá en cuenta sus preferencias, y por ejemplo en el caso de las páginas centrales, devolverá en primer lugar a su favorita. Queremos comprobar que nuestro algoritmo de personalización responde correctamente a un escenario de este tipo. También nos interesa ver como actúa frente a preferencias por sitios de menor importancia.

Realizaremos el mismo experimento con el mismo grafo de entrada previamente detallado, pero con diferentes entradas de personalización. En primer lugar, se utilizará como control el resultado normal de PageRank; luego 3 historiales distintos donde el individuo prefiere cada una de las 3 páginas centrales diferentes; finalmente, un historial donde el usuario preferie mayoritariamente una página arbitraria. Los historiales se encuentran detallados en los historiales 1, 2 y 3. El parámetro *"p"* de personalización elegido

es 0.5.

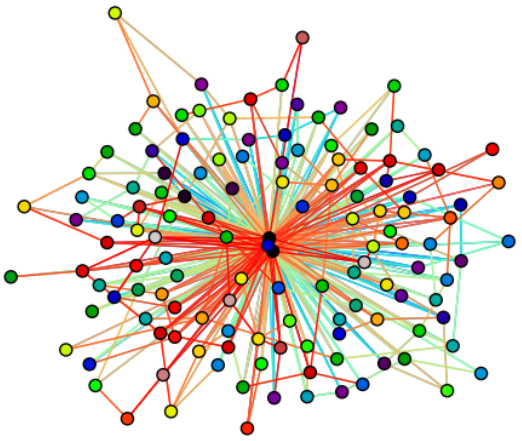


Figura 8: Grafo del experimento 7

Página	Cantidad de veces visitada
5	19
34	11
80	12
129	125

Historial 1: Historial que prioritiza la página 129

Página	Cantidad de veces visitada
5	19
34	11
80	12
130	125

Historial 2: Historial que prioritiza la página 130

Página	Cantidad de veces visitada
5	19
34	11
80	12
131	125

Historial 3: Historial que prioritiza la página 131

Página	Cantidad de veces visitada
5	19
34	11
80	12
124	125

Historial 4: Historial que prioritiza la página 124

3.2.6. Resultados del Experimento 7

Orden	Página
1	130
2	131
3	129
...	—
46	5
...	—
67	124

Tabla 1: Ranking de PageRank sin personalización

Orden	Página
1	129
2	130
3	131
...	—
9	5
...	—

Tabla 2: Ranking de PageRank utilizando el historial que prioritiza la página 129

Orden	Página
1	130
2	131
3	129
...	—
8	5
...	—

Tabla 3: Ranking de PageRank utilizando el historial que prioritiza la página 130

Orden	Página
1	131
2	130
3	129
...	—
15	5
...	—

Tabla 4: Ranking de PageRank utilizando el historial que prioritiza la página 131

Orden	Página
1	130
2	131
3	129
4	5
...	—
17	124
...	—

Tabla 5: Ranking de PageRank utilizando el historial que prioritiza la página 124, con $p = 0{,}5$

Orden	Página
1	5
2	130
3	131
4	129
5	124
...	—

Tabla 6: Ranking de PageRank utilizando el historial que prioritiza la página 124, con $p = 0{,}95$

Orden	Página
1	130
2	131
3	129
4	5
5	124
...	—

Tabla 7: Ranking de PageRank utilizando el historial que prioritiza la página 124, con $p = 0{,}75$ o $p = 0{,}85$

El resultado del control (que se puede observar en la tabla 1) efectivamente tiene a los 3 sitios centrales (en nuestro experimento, páginas 129, 130 y 131) en las primeras 3 posiciones, en el orden 130, 131 y luego 129. El sitio arbitrario (en este caso, el 124) se encuentra en el lugar 67 del ranking. Las 3 iteraciones que prioritizan las distintas páginas centrales (cuyos resultados se pueden encontrar en las tablas 2, 3 y 4) efectivamente tienen como resultado estos sitios en el primer lugar del ranking, que es lo deseado. A su vez, la iteración (resultado en la tabla 5) que prioritiza la página arbitraria la tiene en el lugar 17. Sin embargo, esta posición es más baja de la que esperamos debido a la alta prioridad que se le dio al sitio en el historial, lo que nos lleva a creer que *"p"* en 0.5 puede ser demasiado sutil. Realizamos nuevamente el experimento con *"p"* en 0.85 y 0.95.

Con $p = 0{,}95$ (resultado en la tabla 6) la página 5 (que se encuentra en el historial de entrada, pero con frecuencia relativamente baja) sobrepasa a los sitios centrales. Este resultado es excesivo, por lo que concluiremos que 0.95 es un valor demasiado alto para *"p"*. Tanto para $p = 0{,}85$ como para $p = 0{,}75$ (resultado en la tabla 7) la página 5 se encuentra en el cuarto lugar y la 124 en el quinto; por lo que concluimos que este es un rango aceptable para el parámetro de personalización.

4. Resultados y Discusión: Ligas Deportivas

A continuación vamos a analizar resultados para el algoritmo GeM en el contexto de la Primera Divisón de la Liga Argentina de Fútbol enfocándonos en el ranking de equipos y experimentando sobre variaciones del ranking estándar, obtención del ranking mediante el método GeM y métodos alternatvos. Tomaremos como información de entrada el campeonato 2015 hasta la fecha 26 inclusive.

Contamos con el ranking oficial (estándar) de la liga que consta de ordenar los equipos por puntaje obtenido. Por cada partido ganado un equipo suma 3 puntos, por cada partido empatado obtiene tan solo 1, mientras que no obtiene puntos si fue derrotado. Como se puede ver el ranking no toma como información relevante los goles. Mas adelante veremos que el método GeM se basa fuertemente en los goles pero no diferencia entre victorias y derrotas tan notoriamente como si lo hace el ranking estándar.

4.1. Experimento 1

Como primer experimento vamos a comparar el ranking obtenido tras la aplicación del método GeM y el ranking estándar tomando como entrada los resultados de todos los partidos hasta la fecha 26 inclusive. Creemos que van a notarse diferencias sobre todo por dos motivos:

- El método GeM no tiene en cuenta los empates, resultado que se da con frecuencia en el futbol argentino;
- El método GeM solo se enfoca en los goles, mientras que el método estándar solo se enfoca en quien ganó/empató/perdió.

4.1.1. Resultados del Experimento 1

	Equipo	Puntos
1	Boca Juniors	58
2	San Lorenzo	54
3	Rosario Central	52
4	Racing Club	49
5	River Plate	48
6	Independiente	45
7	Banfield	43
8	Belgrano	43
9	Estudiantes (LP)	42
10	Tigre	42
11	Quilmes	39
12	Lanús	38
13	Unión	38
14	Gimnasia y Esgrima (LP)	37
15	Newell's Old Boys	33
16	San Martín (SJ)	32
17	Aldosivi	30
18	Sarmiento	30
19	Argentinos Juniors	29
20	Olimpo	29
21	Temperley	29
22	Defensa y Justicia	27
23	Huracán	26
24	Vélez Sarsfield	26
25	Godoy Cruz	25
26	Colón	24
27	Arsenal	23
28	Atlético de Rafaela	22
29	Nueva Chicago	17
30	Crucero del Norte	14

Ranking mediante método estándar.

	Equipo	Numero
1	Boca Juniors	0.0860189
2	Aldosivi	0.0653536
3	River Plate	0.0634998
4	San Lorenzo	0.0620352
5	Rosario Central	0.0484731
6	Racing Club	0.0478783
7	San Martín (SJ)	0.0439558
8	Quilmes	0.0423821
9	Newell's Old Boys	0.0382332
10	Vélez Sarsfield	0.0376689
11	Independiente	0.0365646
12	Belgrano	0.0363978
13	Gimnasia y Esgrima (LP)	0.0335849
14	Banfield	0.0307372
15	Estudiantes (LP)	0.0295322
16	Unión	0.0288744
17	Tigre	0.0273356
18	Sarmiento	0.026108
19	Huracán	0.0246666
20	Defensa y Justicia	0.023737
21	Lanús	0.022729
22	Olimpo	0.0224689
23	Arsenal	0.0208702
24	Godoy Cruz	0.017516
25	Temperley	0.016045
26	Crucero del Norte	0.0160393
27	Argentinos Juniors	0.0153984
28	Nueva Chicago	0.0142318
29	Atlético de Rafaela	0.0113878
30	Colón	0.0102765

Ranking mediante método GeM.

En los resultados se puede observar que hay variaciones en las posiciones de los distintos equipos. Esto era de esperarse porque los métodos usados se enfocan en distintos aspectos.

Vamos a analizar el caso particular de Aldosivi. En el ranking estándar se encuentra en la posición 17, consecuencia de tan solo haber ganado 8 partidos y empatar 6 sobre 26 jugados. Mientras que en el ranking GeM se posiciona en el segundo lugar, veamos por qué sucede esto. Recordemos que el método GeM cuenta la diferencia de goles para con su rival en sus partidos ganados. Aldosivi, en los 8 partidos ganados no recibió goles y convirtió 15. Cuenta entonces con una diferencia de goles importante, que lo posiciona en segundo lugar.

Con este claro ejemplo podemos ver la diferencia de enfoques de los dos métodos. Creemos que para un deporte como el futbol el método de GeM no está bien formulado y no se enfoca en los resultados finales. Tal vez en otras ligas donde la diferencia de goles entre el ganador y el perdedor sea mayor (como la española o la alemana) y se den pocos empates pueda acercarse mas al método estándar y/o competir con él sobre cuál debería ser el oficial.

4.2. Experimento 2

Luego de analizar los resultados del experimento 1 queremos ver mas en detalle la influencia de los empates en el ranking obtenido por el método GeM. Sabemos que al no tener en cuenta estos resultados nos alejamos del ranking estándar, pero queremos determinar si es factible que con algún tipo de modificación podamos ajustarlo a un deporte con empates recurrentes (en este caso el fútbol).

Para tomar en consideración a los empates vamos a igualarlos a una victoria 1 a 0: "empate = victoria por 1 a 0". Sabemos que ésto influye positivamente sobre los equipos con muchos empates, pero al darle un resultado favorable mínimo (1 a 0) creemos que no afectará demasiado, manteniendo así la base del enfoque del método GeM y dándole cierta importancia a los empates.

4.2.1. Resultados del Experimento 2

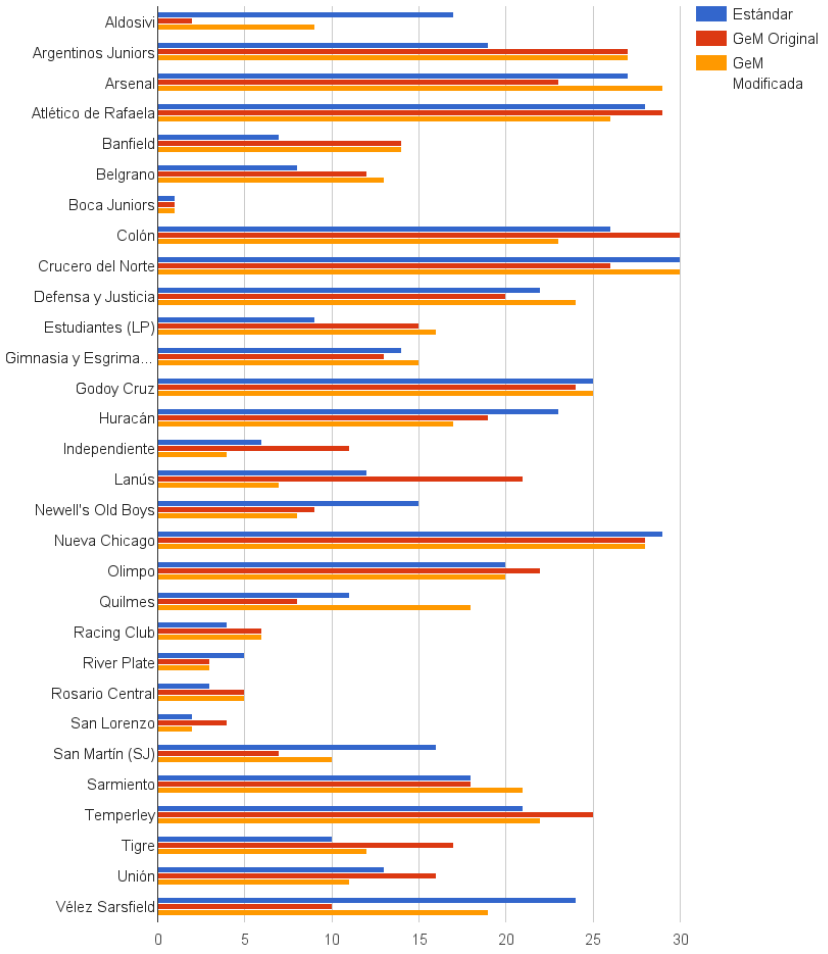


Figura 9: Comparaciones de Rankings

En los resultados obtenidos vemos que la modificación del método GeM generó cambios en el posicionamiento de los equipos con empates en su historial. En algunos casos estos cambios lo alejaron de su posicionamiento en el ranking estándar mientras que en la mayoría de los casos lo acercaron, dándole así un cierto valor al empate que antes no era tenido en cuenta. Como nombramos antes, mantuvimos el enfoque del método GeM pero le agregamos un pequeño valor a los empates que los generó una tabla que creemos mas adecuada a la realidad del fútbol.

	Equipo	**Puntos**
1	Boca Juniors	58
2	San Lorenzo	54
3	Rosario Central	52
4	Racing Club	49
5	River Plate	48
6	Independiente	45
7	Banfield	43
8	Belgrano	43
9	Estudiantes (LP)	42
10	Tigre	42
11	Quilmes	39
12	Lanús	38
13	Unión	38
14	Gimnasia y Esgrima (LP)	37
15	Newell's Old Boys	33
16	San Martín (SJ)	32
17	Aldosivi	30
18	Sarmiento	30
19	Argentinos Juniors	29
20	Olimpo	29
21	Temperley	29
22	Defensa y Justicia	27
23	Huracán	26
24	Vélez Sarsfield	26
25	Godoy Cruz	25
26	Colón	24
27	Arsenal	23
28	Atlético de Rafaela	22
29	Nueva Chicago	17
30	Crucero del Norte	14

Ranking mediante método estándar.

	Equipo	**Numero**
1	Boca Juniors	0.0532173
2	San Lorenzo	0.0466695
3	River Plate	0.0466113
4	Independiente	0.0443246
5	Rosario Central	0.0436734
6	Racing Club	0.0391814
7	Lanús	0.0385408
8	Newell's Old Boys	0.0383858
9	Aldosivi	0.0381275
10	San Martín (SJ)	0.0370988
11	Unión	0.0370056
12	Tigre	0.0347199
13	Belgrano	0.0340198
14	Banfield	0.0340106
15	Gimnasia y Esgrima (LP)	0.0326107
16	Estudiantes (LP)	0.0319726
17	Huracán	0.0312592
18	Quilmes	0.0308084
19	Vélez Sarsfield	0.0306859
20	Olimpo	0.0304997
21	Sarmiento	0.0302557
22	Temperley	0.0277299
23	Colón	0.0267565
24	Defensa y Justicia	0.0262358
25	Godoy Cruz	0.024695
26	Atlético de Rafaela	0.024516
27	Argentinos Juniors	0.023696
28	Nueva Chicago	0.023149
29	Arsenal	0.0230411
30	Crucero del Norte	0.0165022

Ranking obtenido por el método GeM modificado.

Haciendo referencia al experimento 1, donde supimos comparar el ranking estándar del obtenido por el método GeM, podemos sumar los resultados obtenidos con el método GeM modificado. Notamos que las similitudes son mayores a pesar de mantener distancia en los enfoques.

4.3. Experimento 3

En este experimento analizaremos el impacto de la variación del parámetro c sobre los resultados de rankings de competencias deportivas usando GeM. Vimos, por resultados anteriores, que aumentar el c hace que el algoritmo se vuelva más lento, ya que tarda más en converger, pero devuelve resultados más justos. En este experimento nos centraremos en las diferencias entre los resultados y no en el tiempo de ejecución (ver Experimento 3 de la sección de resultados para Páginas Web).

Para llevar a cabo el experimento compararemos las tablas de posiciones obtenidas por cada equipo para ciertos valores de c, comparándolas con la posición actual en el ranking de la AFA.

4.3.1. Resultados del Experimento 3

Observemos la comparación entre las posiciones obtenidas por los equipos según cada c:

Pos. AFA	**Equipo**	**c=0.85**	**c=0.65**	**c=0.45**	**c=0.25**	**c=0.00**
1	Boca Juniors	1	1	1	1	1
2	San Lorenzo	4	3	2	2	1
3	Rosario Central	5	6	6	5	1
4	Racing Club	6	5	5	4	1
5	River Plate	3	2	3	3	1
6	Independiente	11	9	9	8	1
7	Banfield	14	14	13	13	1
8	Belgrano	12	11	10	9	1
9	Estudiantes (LP)	15	15	16	16	1
10	Tigre	17	15	15	14	1
11	Quilmes	8	7	7	7	1
12	Lanús	21	19	18	14	1
13	Unión	16	16	17	18	1
14	Gimnasia y Esgrima (LP)	13	13	12	12	1
15	Newell's Old Boys	9	10	11	11	1
16	San Martín (SJ)	7	8	8	10	1
17	Aldosivi	2	4	4	6	1
18	Sarmiento	18	18	19	19	1
19	Argentinos Juniors	27	26	25	25	1
20	Olimpo	22	23	23	23	1
21	Temperley	25	25	26	26	1
22	Defensa y Justicia	20	21	22	22	1
23	Huracán	19	20	20	20	1
24	Vélez Sarsfield	10	12	14	17	1
25	Godoy Cruz	24	24	24	24	1
26	Colón	30	30	30	30	1
27	Arsenal	23	22	21	21	1
28	Atlético de Rafaela	29	29	29	29	1
29	Nueva Chicago	28	28	28	28	1
30	Crucero del Norte	26	27	27	27	1

Para las fechas del torneo de la AFA, no se ven grandes cambios en los resultados obtenidos variando el c excepto en los casos de Lanús, y Vélez. Esto es una diferencia con el contexto de Páginas Web, ya que como las matrices de Markov de las competencias de fútbol no son esparsas (todos juegan contra todos), a diferencia del caso de las páginas web, no se nota tanto el impacto de la teletransportación. Sin embargo, como vimos en experimentos anteriores, debido a las diferencias de modelado del empate no podemos comparar cualitativamente si los resultados son más justos.

Además, podemos ver que al disminuir el c las distribuciones del vector estacionario se vuelven más equitativas, siendo el caso extremo $c = 0$, donde como se ve en la tabla, todos los equipos terminan en la misma posición.

5. Conclusiones

Concluimos que el algoritmo PageRank es superior al In-Deg para establecer un ranking de páginas web. También concluimos que el tiempo de convergencia depende no sólo de la cantidad de elementos no-nulos de la matriz de transición (es decir, la cantidad de vértices en el grafo que representa a la red), sino de la cantidad de páginas en la web representada (la cantidad de nodos en el grafo). Adicionalmente, concluimos que el tiempo de convergencia aumenta al incrementar el parámetro *"c"*.

También concluimos que nuestra variante del algoritmo PageRank de personalización es efectiva, cuando se emplea un parámetro *"p"* apropiado (alrededor del rango [0.75, 0.85].

Por parte de la aplicación del método GeM en ligas deportivas podemos concluir que es un método interesante para realizar comparaciones con los rankings estándar. Nos deja con ganas de experimentar sobre otras ligas o deportes y ver que resultados obtenemos. Por cuestiones de tiempo no pudimos llevar a cabo estos experimentos, pero sabemos que podemos ir ajustando los parámetros y variando algunos pasos del método para lograr resultados diversos y enfocar el ranking en los datos que creamos importantes.

Pudimos notar que los tiempos de ejecución no fueron relevantes y no tuvimos necesidad de optimizar el proceso ni las estructuras de memoria, dado que los datos eran muy acotados.

En terminos generales pudimos tomar métodos numéricos y aplicarlos sobre casos reales. Esto implicó tener cuidado con los costos (tanto de tiempo como de memoria) y verificar los resultados con respecto a los datos reales. No nos llevamos grandes sorpresas pero nos pareció interesante y entretenido poder aplicar el TP a casos reales y cercanos a la vida diaria y ver como se comportaban.

6. Apéndices

6.1. Apéndice A

Enunciado

El objetivo del trabajo es experimentar en el contexto planteado utilizando el algoritmo PageRank con las variantes propuestas. A su vez, se busca comparar los resultados obtenidos cualitativa y cuantitativamente con los algoritmos tradicionales utilizados en cada uno de los contextos planteados. Los métodos a implementar (como mínimo) en ambos contexto planteados por el trabajo son los siguientes:

1. *Búsqueda de páginas web:* PageRank e In-deg, éste último consiste en definir el ranking de las páginas utilizando solamente la cantidad de ejes entrantes a cada una de ellas, ordenándolos en forma decreciente.

2. *Rankings en competencias deportivas:* GeM y al menos un método estándar propuesto por el grupo (ordenar por victorias/derrotas, puntaje por ganado/empatado/perdido, etc.) en función del deporte(s) considerado(s).

El contexto considerado en 1., en la búsqueda de páginas web, representa un desafío no sólo desde el modelado, si no también desde el punto de vista computacional considerando la dimensión de la información y los datos a procesar. Luego, dentro de nuestras posibilidades, consideramos un entorno que simule el contexto real de aplicación donde se abordan instancias de gran escala (es decir, n, el número total de páginas, es grande). Para el desarrollo de PageRank, se pide entonces considerar el trabajo de Bryan y Leise [4] donde se explica la intución y algunos detalles técnicos respecto a PageRank. Además, en Kamvar et al. [6] se propone una mejora del mismo. Si bien esta mejora queda fuera de los alcances del trabajo, en la Sección 1 se presenta una buena formulación del algoritmo. En base a su definición, P_2 no es una matriz esparsa. Sin embargo, en Kamvar et al. [6, Algoritmo 1] se propone una forma alternativa para computar $x^{(k+1)} = P_2 x^{(k)}$. Este resultado debe ser utilizado para mejorar el almacenamiento de los datos.

En la práctica, el grafo que representa la red de páginas suele ser esparso, es decir, una página posee relativamente pocos links de salida comparada con el número total de páginas. A su vez, dado que n tiende a ser un número muy grande, es importante tener en cuenta este hecho a la hora de definir las estructuras de datos a utilizar. Luego, desde el punto de vista de implementación se pide utilizar alguna de las siguientes estructuras de datos para la representación de las matrices esparsas: *Dictionary of Keys* (dok), *Compressed Sparse Row* (CSR) o *Compressed Sparse Column* (CSC). Se deberá incluir una justificación respecto a la elección que consdiere el contexto de aplicación. Además, para PageRank se debe implementar el método de la potencia para calcular el autovector principal. Esta implementación debe ser realizada íntegramente en C++.

En función de la experimentación, se deberá realizar un estudio particular para cada algoritmo (tanto en términos de comportamiento del mismo, como una evaluación de los resultados obtenidos) y luego se procederá a comparar cualitativamente los rankings generados. La experimentación deberá incluir como mínimo los siguientes experimentos:

1. Estudiar la convergencia de PageRank, analizando la evolución de la norma Manhattan (norma L_1) entre dos iteraciones sucesivas. Comparar los resultados obtenidos para al menos dos instancias de tamaño mediano-grande, variando el valor de c.

2. Estudiar el tiempo de cómputo requerido por PageRank.

3. Para cada algoritmo, proponer ejemplos de tamaño pequeño que ilustren el comportamiento esperado (puede ser utilizando las herramientas provistas por la cátedra o bien generadas por el grupo).

Puntos opcionales:

1. Demostrar que los pasos del Algoritmo 1 propuesto en Kamvar et al. [6] son correctos y computan $P_2 x$.

2. Establecer una relación con la proporción entre $\lambda_1 = 1$ y $|\lambda_2|$ para la convergencia de PageRank.

El segundo contexto de aplicación no presenta mayores desafíos desde la perspectiva computacional, ya que en el peor de los casos una liga no suele tener mas que unas pocas decenas de equipos. Más aún, es de esperar que en general la matriz que se obtiene no sea esparsa, ya que probablemente un equipo juegue contra un número significativo de contrincantes. Sin embargo, la popularidad y sensibilidad del problema planteado requieren de un estudio detallado y pormenorizado de la calidad de los resultados obtenidos. El objetivo en este segundo caso de estudio es puramente experimental.

En función de la implementación, aún cuando no represente la mejor opción, es posible reutilizar y adaptar el desarrollo realizado para páginas web. También es posible realizar una nueva implementación desde cero, simplificando la operatoria y las estructuras, en C++, Matlab o Python.

La experimentación debe ser realizada con cuidado, analizando (y, eventualmente, modificando) el modelo de GeM:

1. Considerar al menos un conjunto de datos reales, con los resultados de cada fecha para alguna liga de alguún deporte.

2. Notar que el método GeM asume que no se producen empates entre los equipos (o que si se producen, son poco frecuentes). En caso de considerar un deporte donde el empate se da con cierta frecuencia no despreciable (por ejemplo, fútbol), es fundamental aclarar como se refleja esto en el modelo y analizar su eventual impacto.

3. Realizar experimentos variando el parámetro c, indicando como impacta en los resultados. Analizar la evolución del ranking de los equipos a través del tiempo, evaluando también la evolución de los rankings e identificar características/hechos particulares que puedan ser determinantes para el modelo, si es que existe alguno.

4. Comparar los resultados obtenidos con los reales de la liga utilizando el sistema estándar para la misma.

Puntos opcionales:

1. Proponer (al menos) dos formas alternativas de modelar el empate entre equipos en GeM.

Parámetros y formato de archivos

El programa deberá tomar por línea de comandos dos parámetros. El primero de ellos contendrá la información del experimento, incluyendo el método a ejecutar (`alg`, 0 para PageRank, 1 para el método alternativo), la probabilidad de teletransportación c, el tipo de instancia (0 páginas web, 1 deportes), el *path* al archivo/directorio conteniendo la definición de la red (que debe ser relativa al ejecutable, o el path absoluto al archivo) y el valor de tolerancia utilizado en el criterio de parada del método de la potencia.

El siguiente ejemplo muestra un caso donde se pide ejecutar PageRank, con una probabilidad de teletransportación de 0.85, sobre la red descripta en `test1.txt` (que se encuentra en el directorio `tests/`), correspondiente a una instancia de ranking aplicado a deportes y con una tolerancia de corte de 0,0001.

```
0 0.85 1 tests/red-1.txt 0.0001
```

Para la definición del grafo que representa la red, se consideran dos bases de datos de instancias con sus correspondientes formatos. La primera de ellas es el conjunto provisto en SNAP [2] (el tipo de instancia es 0), con redes de tamaño grande obtenidos a partir de datos reales. Además, se consideran las instancias que se forman a partir de resultados de partidos entre equipos, para algún deporte elegido por el grupo.

En el caso de la base de SNAP, los archivos contiene primero cuatro líneas con información sobre la instancia (entre ellas, n y la cantidad total de links, m) y luego m líneas con los pares i, j indicando que i apunta a j. A modo de ejemplo, a continuación se muestra el archivo de entrada correspondiente a la red propuesta en Bryan y Leise [4, Figura 1]:

```
# Directed graph (each unordered pair of nodes is saved once):
# Example shown in Bryan and Leise.
# Nodes: 4 Edges: 8
# FromNodeId    ToNodeId
1   2
1   3
1   4
2   3
2   4
3   1
4   1
4   3
```

Para el caso de rankings en ligas deportivas, el archivo contiene primero una línea con información sobre la cantidad de equipos (n), y la cantidad de partidos totales a considerar (k). Luego, siguen k lńeas donde cada una de ellas representa un partido y contiene la siguiente información: número de fecha (es un dato opcional al problema, pero que puede ayudar a la hora de experimentar), equipo i, goles equipo i, equipo j, goles equipo j. A continuación se muestra el archivo de entrada con la información del ejemplo utilizado en Govan et al. [5]:

```
6 10
1 1 16 4 13
1 2 38 5 17
1 2 28 6 23
1 3 34 1 21
1 3 23 4 10
1 4 31 1 6
1 5 33 6 25
1 5 38 4 23
1 6 27 2 6
1 6 20 5 12
```

Es importante destacar que, en este último caso, los equipos son identificados mediante un número. Opcionalmente podrá considerarse un archivo que contenga, para cada equipo, cuál es el código con el que se lo identifica.

Una vez ejecutado el algoritmo, el programa deberá generar un archivo de salida que contenga una línea por cada página (n líneas en total), acompañada del puntaje obtenido por el algoritmo PageRank/In-deg/método alternativo.

Para generar instancias de páginas web, es posible utilizar el código Python provisto por la cátedra. La utilización del mismo se encuentra descripta en el archivo README. Es importante mencionar que, para que el mismo funcione, es necesario tener acceso a Internet. En caso de encontrar un bug en el mismo, por favor contactar a los docentes de la materia a través de la lista. Desde ya, el código puede ser modificado por los respectivos grupos agregando todas aquellas funcionalidades que consideren necesarias.

Para instancias correspondientes a resultados entre equipos, la cátedra provee un conjunto de archivos con los resultados del Torneo de Primera División del Fútbol Argentino hasta la Fecha 23. Es importante aclarar que los dos partidos suspendidos, River - Defensa y Justicia y Racing - Godoy Cruz han sido arbitrariamente completados con un resultado inventado, para simplificar la instancia. En función de datos reales, una alternativa es considerar el repositorio DataHub [1], que contiene información estadística y resultados para distintas ligas y deportes de todo el mundo.

7. Referencias

Referencias

[1] Datahub. `http://datahub.io`.

[2] Stanford large network dataset collection. `http://snap.stanford.edu/data/#web`.

[3] 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, April 1998.

[4] Kurt Bryan and Tanya Leise. The linear algebra behind google. *SIAM Review*, 48(3):569–581, 2006.

[5] Angela Y. Govan, Carl D. Meyer, and Rusell Albright. Generalizing google's pagerank to rank national football league teams. In *Proceedings of SAS Global Forum 2008*, 2008.

[6] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, and Gene H. Golub. Extrapolation methods for accelerating pagerank computations. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 261–270, New York, NY, USA, 2003. ACM.

[7] Taher H. Haveliwala and Sepandar D. Kamvar The Second Eigenvalue of the Google Matrix Stanford University Technical Report, 2003.