El alumno estratégico: una aplicación del algoritmo PageRank

Alberto Jiménez y Mercedes de la Torre

May 15, 2015

Abstract

En este trabajo se describirá el algoritmo PageRank y se utilizará para *rankear* las materias del Plan Conjunto de Economía y Matemáticas Aplicadas que se ofrece en el Instituto Tecnológico Autónomo de México (ITAM). Se definirá el *Avance Académico* en función del rango de cada materia y se ilustrará su utilidad en la toma decisiones por medio de una simulación por agentes.

Una de las principales tareas de un motor de busqueda es decidir el orden en el cual se presentan los resultados al usuario con base en a) la relevancia de las páginas en función de la busqueda y b) la relevancia general de la página. PageRank es el algortimo que desarrollaron Larry Page y Sergey Brin (fundadores de Google) para realizar esta tarea.

Nuestro objetivo es describir el algoritmo y demostrar su utilidad en la toma de decisiones de un alumno del ITAM que cursa el Plan Conjunto de Econoía y Matemáticas Aplicadas, y utilizarlo para redefinir el concepto tradicional de *Avance Académico*.

El algoritmo PageRank

Los motores de busqueda realizan diversas tareas. Aquí están las más importantes.

1. Localizar páginas web y almacenar información pertinente.
2. En respuesta al *query* del usuario, ejecutar busquedas de las páginas relevantes en tiempo real.
3. Decidir el orden en el cual se presentan los resultados al usuario con base en
   a) La relevancia de las páginas en función de la busqueda.
   b) La relevancia general de la página.

El PageRank se encarga de la tercera tarea. Su objetivo es medir la importancia de cada página en la *web*: informar al usuario el punto 3b.

Para dar una descripción intuitiva del algoritmo, supongamos que la web es una matriz de $4x4$. En la Tabla 1 se muestra un modelo simplificado de la *web*. Supongamos que tenemos 4 páginas

Table 1: Entradas de la matriz de adyacencia. En la última columna se presenta el grado de cada página

	MATEMÁTICAS	FÍSICA	LETRAS	ECONOMÍA	grado
MATEMÁTICAS	0	1	1	1	3
FÍSICA	1	0	0	0	1
LETRAS	1	0	0	0	1
ECONOMÍA	1	0	1	0	2

MATEMÁTICAS,FÍSICA,ECONOMÍA y LETRAS. Generalmente *W(i,j)* es igual a 1 si existe un *link* de la página i a la página j y zero en otro caso.

El *grado* de la página i se define como la suma de las entradas del renglón i de la **matriz de adyacencia**.

Nos apoyaremos en la siguiente notación para generalizar el concepto.

- Sea N el número total de páginas.
- Sea W la **matriz de adyacencia** de N X N, de tal manera que w_{ij} es igual a 1 si existe un *link* de la página i a la página j, cero en otro caso.
- Sea deg_i el grado del reglón i, de tal manera que $deg_i := \sum_{j=1}^{N} w_{ij}$.

En el trabajo original [1] se asume que $deg_i \neq 0$ para toda $1 \leq i \leq N$, sin embargo, es un supuesto que no podremos manterner en nuestro trabajo. Para fines de este apartado, tomaremos el supuesto como válido.

El algoritmo *PageRank* procede de manera iterativa, asignando un valor r_j^n a la página j en la iteración n.La iteración es

$$r_j^n = (1-d) + d\sum_{i=1}^{N} \frac{w_{ij} r_i^{n-1}}{deg_i} \tag{1}$$

Donde d representa una constante tal que $0 < d < 1$. Se utilizará $d = 0.85$. La clave del algoritmo es entender que un *link* de i a j es un voto de confianza para la página j de la página i. Por lo tanto, la importancia de la página j se puede medir con *links* referenciados a ésta. Se ponderan los votos de acuerdo a la importancia de los votantes: un voto de una página importante pesa más que uno de otra menos importante. También se escala cada voto por la cantidad de votos que emite el votante. Esto significa que escalamos los pesos w_{ij} con su respectivo grado, deg_i, de tal manera que tenemos $\frac{w_{ij}}{deg_i}$. El último elemento del algoritmo es la constante d. A cada nodo se le da una importancia de $1-d$ de manera gratuita, y después se le asigna d veces el valor de los votos de confianza.

En resumen, la iteración (1) para actualizar el rango de la página j se puede describir de la siguiente manera:

- Por cada página i que vote por j, suma el valor escalado del *ranking* actual,$\frac{r_i^{n-1}}{deg_i}$.
- Multiplicamos por la constante d y sumamos $1-d$.

Ranking del Plan Conjunto de Economía y Matemáticas Aplicadas

El Instituto Tecnológico Autónomo de México (ITAM) ofrece la oportunidad de cursar dos programas de manera simultanea aprovechando las materias en común que tiene cada licenciatura, a esta modalidad se le conoce como Plan Conjunto. En este trabajo *rankearemos* las materias del Plan Conjunto de Economía y Matemáticas Aplicadas (Eco y Mate).

El Plan Conjunto de Eco y Mate está compuesto por 64 materias que se cursan en once semestres intensivos. De estas 64 materias, 60 ya están definidas y 4 son elegidas libremente por el alumno. Dado que no están definidas las optativas, **supondremos que el plan de Eco y Mate consiste en 60 materias**, las cuales se muestran en la Tabla 2.

Table 2: Materias del Plan Conjunto de Economía y Matemáticas Aplicadas

Indice	Materia
1	Introducción a la Matemática Superior
2	Herramientas Computacionales y Algorítmicas
3	Economía I
...	...
...	...
...	...
58	Economía Internacional
59	Seminario de la Investigación Económica I
60	Seminario de la Investigación Económica II

La **matriz de adyacencia** del plan de estudios de Eco y Mate se construye de la siguiente manera.

Si la clase i es prerrequisito para la clase j, entonces $w_{ij} = 1$. Aquí, el voto de confianza es otorgado a aquellas materias que utilizan el contenido de las materias que emiten su voto, de tal manera que el $PageRanking$ se puede interpretar como encontrar las materias más "difíciles" o "densas".

El supuesto $deg_i \neq 0$: dangling pages

El algoritmo $PageRank$, definido como en la sección anterior, se sostiene de un supuesto muy fuerte: $deg_i \neq 0$. Este supuesto no se cumple en nuestra **matriz de adyacencia**, por lo que tendremos que agregar *links* artificiales. Si bien agregar *links* artificiales es el método más utilizado, no es el más adecuado. En [2] se desarrolla un algoritmo cuya eficiencia incrementa entre más *dangling pages* se tengan.Dado que nuestro objetivo no es *rankear* las materias de Eco y Mate sino analizar como un agente toma decisiones dado que sabe que una materia tiene mayor rango que otra (sea cual sea la materia), utilizaremos el método de *links* artificiales para satisfacer la condición $deg_i \neq 0$

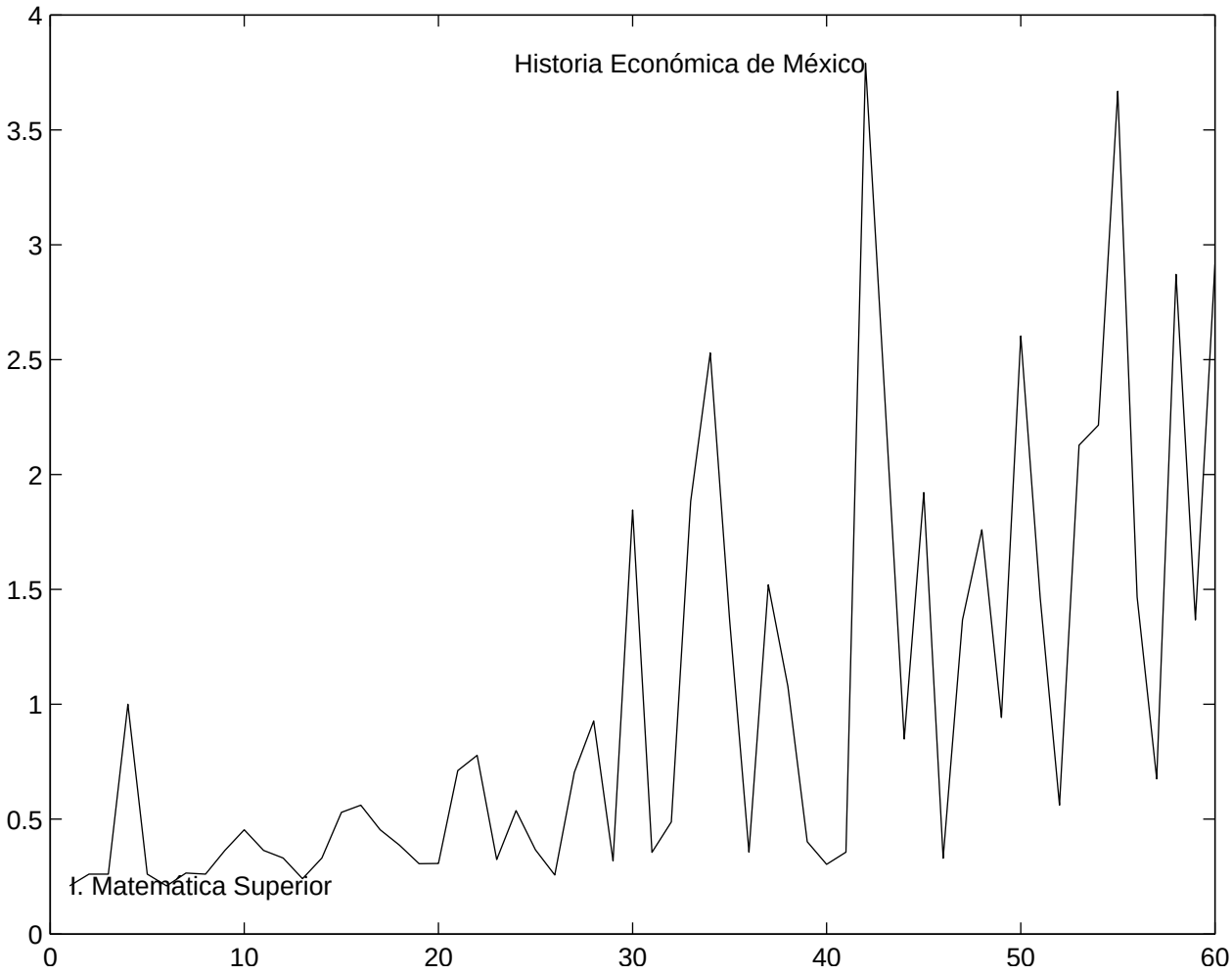

Figure 1: Rango de las materias de Eco y Mate

Ranking

Los resultados del *PageRank* se muestran en el Figura 1. Se puede observar que las materias de los últimos semestres , tomando en cuenta que mayor índice implica que la materia se cursa en un semestre más avanzado, tienen un mayor rango. Esto no es sorprendente, dado que las materias de últimos semestres son las que utlizan el contenido de todas las anteriores. *Historia Económica de México* es la materia de mayor rango en el plan de estudios, una explicación es que es referenciada por dos de las materias más relevantes en todo el plan de estudios: *Economía V y Problemas de la Realidad Mexicana Contemporánea*.

El estratégico, el torpe y el algoritmo

El objetivo del *PageRank* es presentar al usuario las páginas más relevantes de acuerdo a su busqueda. En este caso lo que se presenta al usuario son materias en función de su relevancia. Cada periodo el agente (nuestro usuario en la simulación) acredita m materias, las cuales son prerrequisito

de otra serie de materias que se podran elegir para cursar en el siguiente periodo. Al conjunto de materias que el agente puede elegir en el periodo t para cursar en el $t+1$ lo llamaremos *conjunto elegible*. El objetivo de la simulación será analizar la utilidad del *PageRank* para la toma de decisiones de nuestro agente. Definiremos distintos tipos de agentes los cuales tienen una meta en común: maximizar su *Avance Academico*.

Simulación

Tenemos dos características que definen a cada agente:

Buen alumno/Mal alumno: Se define en función de la probabilidad que tiene el agente de acreditar una materia. Sea p es la probabilidad de acreditar una materia, el **Buen alumno** será aquel que tenga $p > .6$

Estratégico/Torpe: Un alumno **Estratégico** será aquel que elija las materias del *conjunto elegible* de acuerdo al *vector de rangos*. Un alumno **Torpe** elegirá las materias del *conjunto elegible* de manera aleatoria.

Estas dos caracteriísticas producen cuatro tipos de agentes

Table 3: Agentes

Buen alumno/Estratégico	Buen alumno/Torpe
Mal alumno/Estratégico	Mal alumno/Torpe

El agente únicamente está interesado en su *Avance Academico*

$$AvanceAcademico = \frac{\sum_{k=1}^{t} R_k}{\sum_{k=1}^{n} R_k} \tag{2}$$

donde

$$R_k = \sum_{k=1}^{m} r_k^j \tag{3}$$

R_k es la suma de los *rangos* de las m materias acreditadas en el periodo k.

La simulación procede como sigue:

- Para cada semestre t el agente debe elgir 6 materias del *conjunto elegible*. Las materias en $t = 0$ ya están determinadas.
- El *conjunto elegible* se define como aquellas materias para las cuales ya se cumple con el prerrequisito.
- Se evaluará el *Avance Academico* de los cuatro agentes en un periodo $t = T$

La simulación se realizará en *Python*. Primero definiremos las funciones sobre las cuales se construirá la simulación.

Listing 1: Suma de rangos de las m materias acreditadas en el periodo k (Eq. 3)

```
def R_k (k,r_k, m, p):
    suma=0
    for j in xrange (1,m):
        if acreditar(p)==1:
            suma+=r_k**j
        else :
            suma+=0
    return suma
```

Listing 2: Avance Académico (Eq. 2)

```
def avance (t, n, m, p):
    suma_t=0
    suma_n=0
    for k in xrange(1,t+1):
        suma_t+= R_k(k, r_k, m, p)
    for l in xrange (1, n+1):
        suma_n+= R_k(l,r_k,m, p)
    return suma_t/suma_n
```

Listing 3: Aprobación de materias

```
#Aqui es donde requerimos nuestra herramienta (Bernoulli)
#llamada al principio.
#Como tenemos diferentes tipos de alumnos,
#especialmente buen/ mal alumno con probabilidad p de
#acreditar cada materia,
#lo que se hace en esta funcion es que metiendo dicha
#probabilidad, nos va a regresar 0 o 1.
#Si nos regresa cero es que no acredito la materia,
#si nos regresa 1 es que si la acredito.

def acreditar(p):
    return bernoulli.rvs(p)

}

#Ahora para descartar las materias ya acreditadas
#para obtener un arreglo de las que faltan
#por acreditar definimos:
def primer_semestre(p):
```

```
    primer=np.zeros(6)
    for i in xrange(0,6):
        if acreditar(p)==1:
            a[i]=0
        else:
            a[i]=1
    return a

#Primero como las primeras seis materias no son
#elegibles vemos si  stas ya fueron acreditadas o no.
# Por lo que creamos un arreglo de 6 entradas con puro ceros,
#y si la acredit  se marca con uno y si no,
#es con cero.
def materias_por_acreditar(matriz, p):
    a=np.zeros(60)
    for i in xrange(0,61):
        if acreditar(p)==1:
            a[i]=0
        else:
            a[i]=1
    return a
```

Listing 4: Conjunto de materias elegibles

```
#Para esta funcion lo que se hizo primero fue definir un arreglo
#de 60 entradas de ceros inicialmente (una por cada materia),
#lo que representa que no se ha acreditado nada
#Para esto vamos a ver si ya la acredito o no, primero
#debemos de saber si ya acredito alguna
#de las que no pudo elegir, por lo que pasamos el arreglo
#del primer semestre a las primeras 6 entradas
#de nuestro arreglo con todas las materias.
#Luego lo que se hizo fue checar el semestre en el que se
#encuentra    t    y como idealmente ya pudo haber llevado t*6
#materias en el mejor de los casos,
#se van a checar esa cantidad de materias.
#Para esto debemos de saber si ya se abrio dicha materia,
#lo cual lo vemos con nuestra matriz. En caso de que si se
#haya acreditado alguna de sus prerrequisitos
#vemos si se pudo acreditar esta nueva materia,
#en cuyo caso ponemos un uno en nuestro arreglo.
#En cualquier otro caso ponemos un cero, ya sea porque
#no se pudo abrir o porque no se acredito.
#Por lo que nuestras materias por acreditar son las
#que se encuentran en cero en este arreglo
```

```
def materias_elegibles(matriz, p):
    b=materias_por_acreditar(matriz, p)
    elegibles=np.zeros(60)
    for i in xrange(0,60):
        for j in xrange (i,60):
            if b[i]==0:
                if matriz[i,j]==1:

                    elegibles[j]=1

    return elegibles

#Para saber que tipo de materias son elegibles despu s de
#haber acreditado sus prerrequisitos,
#creamos otro arreglo con ceros y mandamos llamar
#nuestras materias que faltan por acreditar en el semestre t.
#Recorremos nuestra matriz para ver si cuales son prerrequisito
#de las materias recorro
#sobre todas las materias para ver cu les abren, pero como
#no va a abrir ninguna materia anterior a  sta ,
#s lo vamos a checar el tri ngulo superior de la matriz y optimizar tiempo.
Con nuestro arreglo llamado    b    que es el que se defini  como el que
#ya tiene las materias acreditadas
#cuenta con un    1    en sus entradas, vamos a revisar que materias abre
#y ponemos en el lugar de esa materia un uno como que ya se abri  en
#nuestro arreglo de elegibles. Idealmente ser a que se abrieran todos los prerre
# pero esto ya se empexo a complicar mucho y quebro el codigo,
#porque hab a muchos supestos detr s de eso.
# Esta funci n me devuelve un arreglo con ceros donde estan las materias
#ya acreditadas o las que no se me han abierto.

#Para siguientes funciones vamos a necesitar el numero de materias que
#puedo escoger para el siguiente semestre,
# por lo que se definio la funci n:

def numero_materias_elegibles(matriz,p):
    suma=0
    for i in xrange(0,60):
        suma+=materias_elegibles(matriz,p)
    return suma
```

Listing 5: El alumno torpe

```
#Lo primero que se hizo en esta funcion fue definir como n el numero
#de materias que puede elegir
#ya que se hayan acreditado sus prerrequisitos.
```

```
#El arreglo  materias_por_escoger  son las materias elegibles,
#el conjunto de eleccion con n entradas va a contener el numero
#del lugar donde se encuentran las materias elegibles,
#y por ultimo materias_elegidas_torpe son las materias que
#elige aleatoriamente el torpe en un semestre.
#Primero recorro mi materias por escoger, si este arreglo tiene 1
#en una entrada entonces si puedo escogerla,
# por lo que el lugar donde se encuentra lo guardamos en e
#l conjunto de materias elegibles.
# Ahora para elegir las marerias al azar, creo una copia del arreglo
#que tiene el conjunto de eleccion pero con
#los numeros en las entradas desordenados
 ( azar=np.random.permutation(conjunto_de_eleccion) .
#Sin perdida de generalidad puedo escoger las primeras seis materias
#que ya puedo escoger y se permutaron en
#el arreglo de azar y las pongo como las materias elegidas por el torpe.
# Esta funcion devuelve el numero de la materia que va a escoger para
#el siguente semestre suponiendo que va a meter 6

def eleccion_alumno_torpe(matriz,p):
    n= numero_materias_elegibles(matriz,p)
    materias_por_escoger=materias_elegibles(matriz,p)
    conjunto_de_eleccion=np.zeros(n)
   #va a contener el numero del lugar donde se
   #encuentran las materias
   #que ya se pueden escoger en el arreglo de
   #materias elegibles
    materias_elegidas_torpe=np.zeros(6)
   #se escogen 6 materias en total
    for k in xrange(0,n):
        for i in xrange(0,60):
            if materias_por_escoger[i]==1:
    #si puedo escoger la materia entonces voy a poner su numero de lugar
    #en un nuevo arreglo llamado conjunto_de_eleccion
                conjunto_de_eleccion[k]=i+1
    #Para que las materias las cuente a partir de 1 y no de cero
    azar=n
p.random.permutation(conjunto_de_eleccion)
   #para elegir las materias al azar creo una copia del arreglo
   # de conjunto de eleccion pero con los numeros desordenados
    for r in
 xrange(0,6):
            materias_elegidas_torpe[r]=azar[r]
            #sin perdida de generalidad puedo escoger las primeras
            #seis materias que ya puedo escoger y se permutaron
            #en el arreglo de azar
    return materis_elegidas_torpe
```

```
 #devuelve el numero de la materia que va a escoger
# para el siguente semestre
    #suponiendo que va a meter 6
```

Resultados

To be continued . . .

References

[1] Higham, Desmond J., and Taylor, Alan, "The Sleekest Link Algorithm," August 2003.

[2] Ipsen, Ilse C.F., Selee,Teresa M.,"PageRank Computation, with Special Attention to Dangling Nodes, " *Siam Journal on Matrix Analysis and Applications*,vol. 29,no. 4, pp. 1281-1296, 2007.