

Trabajo fin de grado

# **Método de ranking en el diseño de un sistema de acceso a la información**

Johanna Capote Robayna

Doble Grado en Ingeniería Informática y Matemáticas



**UNIVERSIDAD  
DE GRANADA**

Tutoras:

Silvia Acid Carrillo

Margarita Arias López

# Índice

<b>1.</b>	<b>Resumen</b>	<b>4</b>
<b>2.</b>	<b>Extended Abstract</b>	<b>5</b>
<b>3.</b>	<b>Introducción</b>	<b>6</b>
<b>4.</b>	<b>Introducción al modelo</b>	<b>7</b>
<b>5.</b>	<b>Descripción del problema</b>	<b>9</b>
<b>6.</b>	<b>Planificación</b>	<b>10</b>
<b>I.</b>	<b>Matemáticas</b>	<b>12</b>
<b>7.</b>	<b>Modelo matemático</b>	<b>12</b>
<b>8.</b>	<b>Teorema de Perron-Frobenius</b>	<b>16</b>
8.1.	Matrices positivas . . . . .	20
8.2.	Matrices no negativas . . . . .	25
8.3.	Enunciado del teorema . . . . .	28
8.4.	Demostración del teorema . . . . .	28
8.5.	Método de las potencias . . . . .	31
<b>II.</b>	<b>Informática</b>	<b>33</b>
<b>9.</b>	<b>Esquema del proyecto</b>	<b>33</b>
<b>10.</b>	<b>Modificaciones previas</b>	<b>34</b>
<b>11.</b>	<b>Pseudocódigo</b>	<b>37</b>
<b>12.</b>	<b>Ejemplos</b>	<b>38</b>
<b>13.</b>	<b>Conjunto de datos</b>	<b>42</b>
<b>14.</b>	<b>Modelos de Recuperación de Información</b>	<b>44</b>
14.1.	Modelo binario . . . . .	44

14.2. Modelo vectorial . . . . .	46
14.3. Técnicas de modificación de la consulta . . . . .	49
<b>15. Interfaz gráfica</b>	<b>52</b>
<b>16. Esquema de las consultas</b>	<b>59</b>
<b>17. Resultados</b>	<b>61</b>
<b>18. Conclusiones y trabajos futuros</b>	<b>72</b>
<b>19. Sistema basado en la información</b>	<b>73</b>
<b>Referencias</b>	<b>74</b>

## 1. Resumen

El objetivo de este trabajo es estudiar y entender el funcionamiento del algoritmo PageRank, además se implementa un buscador, cuya base es este algoritmo, que a través de consultas consiga recuperar información útil para el usuario.

En cuanto a la parte matemática se ha realizado un estudio del modelo utilizado en dicho algoritmo adaptándolo a un problema de valores y vectores propios. Se ha estudiado las propiedades de las matrices y los requisitos que deben cumplir estas para que nuestro problema tenga solución. Además se han estudiado métodos numéricos como el método de las potencias, algoritmo clave para calcular el vector solución de nuestro problema, ya que se utiliza una cantidad de datos considerablemente grandes.

Para la parte de informática se ha implementado un buscador en lenguaje python, que consigue ordenar los archivos de la conjunto de datos siguiendo el algoritmo PageRank. Adicionalmente se ha implementado una interfaz gráfica para mostrar con mayor comodidad los resultados del algoritmo. Para poder realizar consultas en el sistema se ha estudiado e implementado los diferentes modelos de la Recuperación de Información, tanto el modelo booleano como el modelo vectorial. Por último, con el objetivo de realizar consultas personalizadas se ha estudiado e implementado la pseudo-realimentación de consulta, la cual extiende la consulta dependiendo de los documentos que le hayan resultado relevantes al usuario que realiza la consulta.

**Palabras clave:** PageRank, Perron-Frobenius, método de las potencias, Recuperación de Información, buscador.

## **2. Extended Abstract**

### 3. Introducción

Fue en la década de los 80 cuando Internet empezaba a darse a conocer con unas primeras páginas web <sup>1</sup> de estilo muy básico y sencillo que tardaban mucho en cargarse. Con el transcurso del tiempo la Red fue creciendo a pasos agigantados almacenando un gran volumen de información lo que provocó un sinnúmero de procesos y cambios en las formas de gestionarla, y es en ese contexto donde surgen los buscadores para dar respuesta a la necesidad de clasificar y gestionar gran cantidad de información lo más rápido posible.

Un buscador o motor de búsqueda es un sistema de recuperación de información, donde el usuario introduce un conjunto de términos <sup>2</sup> o palabras clave y el buscador le devuelve una lista de resultados <sup>3</sup> ordenados bajo una serie de criterios. WebCrawler, Lycos, AltaVista o Yahoo entre otros, son los primeros buscadores de la Historia que funcionaban de una forma muy similar, rastreaban la Red y clasificaban las páginas en función de las veces que contenían las palabras clave introducidas. Eran los más utilizados en esa época hasta que en 1998 son desbancados por Google, el mayor motor de búsqueda de todos los tiempos, que consigue imponerse al resto de buscadores gracias a un nuevo algoritmo de búsqueda (PageRank), capaz de ordenar una gran cantidad de información en poco tiempo.

PageRank es un algoritmo basado en el Álgebra lineal, Teoría de Grafos y Probabilidad, fue diseñado por Sergey Brin y Lawrence Page, el primero graduado en Matemáticas y el segundo en Informática y ambos estudiantes de doctorado de Informática de la Universidad de Stanford. La pregunta que buscaban resolver era clara “¿en qué orden mostrar los resultados de la búsqueda?”. En 1997 cuando Brin y Page empezaron a trabajar en el diseño del algoritmo se plantearon como objetivo principal que en el gran número de los casos, al menos una de las primeras páginas que se muestran como resultado contenga información útil para el usuario. Hay que tener en cuenta que en esta época Internet no era tan grande como ahora, habían censadas en torno a 100 millones de páginas web y los buscadores más famosos de la época eran capaces de atender 20 millones de consultas al día, mientras que hoy en día Google es capaz de atender a 200 millones de consultas diarias. Pasaremos a explicar cómo el algoritmo de PageRank es capaz de ordenar estas millones de páginas web en tan poco tiempo.

---

<sup>1</sup>En el contexto de R.I. se suelen denominar documentos.

<sup>2</sup>En el contexto de R.I. se suele denominar consulta (*query*).

<sup>3</sup>En el contexto de R.I. los resultados son los documentos o páginas relevantes de la consulta.

## 4. Introducción al modelo

Lo primero que hace Google, incluso antes de que el usuario realice la búsqueda es organizar el contenido de Internet ayudándose de un índice. Este índice se va actualizando y va añadiendo páginas nuevas y modificando la información de las páginas ya existentes en un proceso llamado “rastreo”, en el cual adquiere información de esas páginas incluyendo los enlaces hacia otras páginas. El índice se ordena con el algoritmo de **PageRank**, se explicará a continuación, que ordena la lista asignando un PageRank más alto a las páginas que calificará de “más interesantes” y un PageRank bajo a las páginas “menos interesantes”.

En un buscador cuando el usuario introduce su búsqueda esta pasa por varias fases:

- **Análisis de las palabras del usuario.** En esta fase se busca entender el significado de la búsqueda, para ello se analiza las palabras usadas y se interpreta lo que el usuario quiere decir con ellas. Este proceso incluye varios pasos como **interpretar los errores de ortografía** o entender el significado que el usuario le está dando a una palabra con varias acepciones ayudándose de un **sistema de sinónimos**. Otro aspecto importante de esta fase es el **algoritmo de novedades**, el cual deduce que en ciertas búsquedas debe mostrar la información más actual, como por ejemplo si el usuario busca “horóscopo” o “resultados del fútbol club Granada” se está interesando por las últimas publicaciones.
- **Búsqueda de coincidencias.** En esta fase se busca entre las páginas del índice aquellas con información coincidente con la búsqueda. Se analiza la frecuencia con la que aparecen las palabras introducidas en las páginas web candidatas y su localización; es decir, si aparecen en el título, encabezamientos o en el cuerpo del texto. Además existen otros algoritmos que analizan si las páginas candidatas pueden resultar interesantes para el usuario, ya que si por ejemplo el usuario busca “pájaros” lo más probable es que no quiera encontrar una página con poca información relevante sobre el tema pero en la que se repite muchas veces la palabra “pájaro”.
- **Mejora del posicionamiento de las páginas útiles.** Una vez obtenidas las páginas web con información potencialmente válida se pasa a clasificar su utilidad con el objetivo de mostrar primero las páginas más útiles para el usuario. En esta fase se analizan varios factores como el número de veces que aparece el término buscado

#### 4. Introducción al modelo

en la página, la fecha de publicación y la calidad de experiencia del usuario en la página. Además se descartan aquellos sitios web con contenido fraudulento o *spam* y se potencia las páginas que en consultas similares hayan sido fiables.

- Devolver los mejores resultados. En esta parte se busca la diversidad de respuestas. Antes de mostrar el resultado de la búsqueda se analiza el resultado en conjunto, se evalúa si hay muchas páginas centradas en la misma interpretación de la búsqueda o si solo existe un tema en los resultados.
- Análisis del contexto. Esta fase no se realiza siempre, en ella se busca “personalizar” la búsqueda con datos como la ubicación o historial de búsqueda. Con ello se consigue reducir la búsqueda a su entorno, por ejemplo si el usuario es español y busca “baloncesto” le saldrá antes información sobre el baloncesto en España que sobre el de otros países o si el usuario busca “conciertos cerca de mí” aparecerán los conciertos más cercanos a su residencia. Además mediante el historial de búsqueda personal el buscador consigue mejorar la interpretación de la búsqueda, por ejemplo si el usuario ha buscado anteriormente “Granada vs Osasuna” y más adelante busca “Granada” puede considerar que el usuario se refiere al equipo de fútbol y no a la ciudad.



## 5. Descripción del problema

El problema que se va a abordar en este trabajo es conseguir un método de *ranking* para ordenar la información según el interés del usuario. Para ello se investigarán las técnicas utilizadas por uno de los mayores motores de búsqueda, Google, y se aplicarán sobre un conjunto de datos específico PMSC-UGR para realizar la parte experimental, que se detallará en la sección 13. Para abordar el problema nos centramos en la parte preliminar a una búsqueda. Como ya se comentó anteriormente, antes de que el usuario realice una búsqueda en Google las páginas web ya se encuentran previamente ordenadas por el algoritmo PageRank. Este algoritmo es en el que se centrará este trabajo dividido en dos grandes partes:

- En la primera parte se realizará un estudio teórico del algoritmo PageRank, en el que se detallará la base matemática de manera formal. En primer lugar se ajustará nuestro problema a un problema de valores y vectores propios y a continuación se desarrollarán varios resultados matemáticos que garantizan la existencia de una solución.
- En la segunda parte se realizará un estudio práctico del algoritmo PageRank sobre el conjunto de datos comentado, además se utilizan distintos modelos de Recuperación de Información en los cuales a partir de consultas el usuario podrá obtener documentos que le resulten interesantes.

## 6. Planificación

En esta sección mostramos las distintas tareas desarrolladas durante el proyecto y su duración.

- Documentación: 2 semanas.
- Marco teórico: 2 semanas.
- Estudio del método de las potencias: 2 semanas.
- Implementación del método de las potencias: 6 semanas.
  - Implementación: 2 semanas.
  - Pruebas: 2 semanas.
  - Dudas: 2 semanas.
- Implementación de la lectura de datos: 2 semanas.
  - Implementación: 1 semana.
  - Pruebas y dudas: 1 semana.
- Estudio teórico de las propiedades de las matrices positivas: 2 semanas.
- Implementación del PageRank: 2 semanas.
  - Implementación: 1 semana.
  - Pruebas y dudas: 1 semana.
- Estudio teórico de las matrices no negativas: 2 semanas.
- Implementación del modelo booleano: 4 semanas.
  - Implementación: 2 semanas.
  - Pruebas y dudas: 2 semanas.
- Estudio teórico del Teorema de Perron-Frobenius: 2 semanas.

## 6. Planificación

- Implementación de la interfaz gráfica: 2 semanas.
  - Implementación: 1 semana.
  - Pruebas y dudas: 1 semana.
- Implementación del modelo vectorial: 2 semanas.
  - Implementación: 1 semana.
  - Pruebas y dudas: 1 semana.

La planificación temporal que se ha seguido en este trabajo es la siguiente:

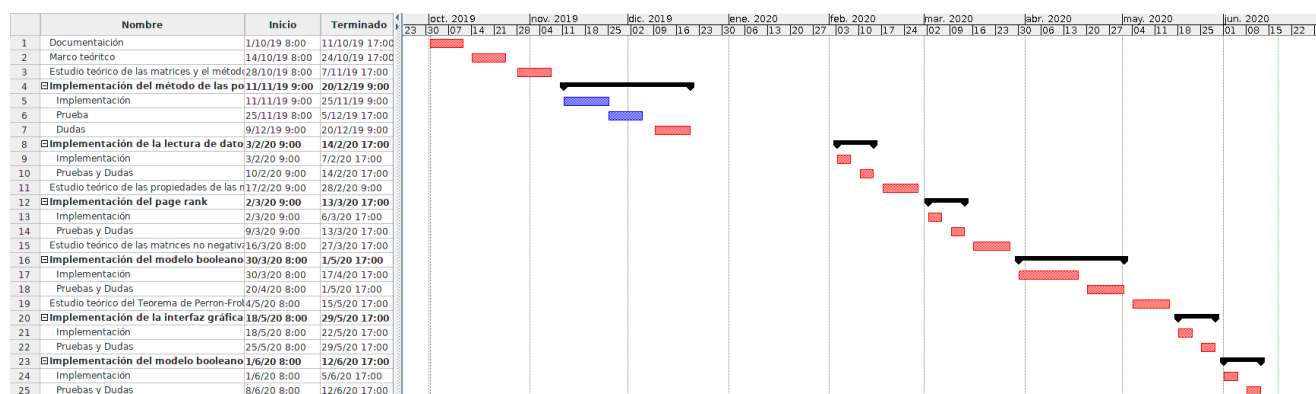


FIGURA 1: Diagrama de Gantt del proyecto.

# Parte I.

## Matemáticas

### 7. Modelo matemático

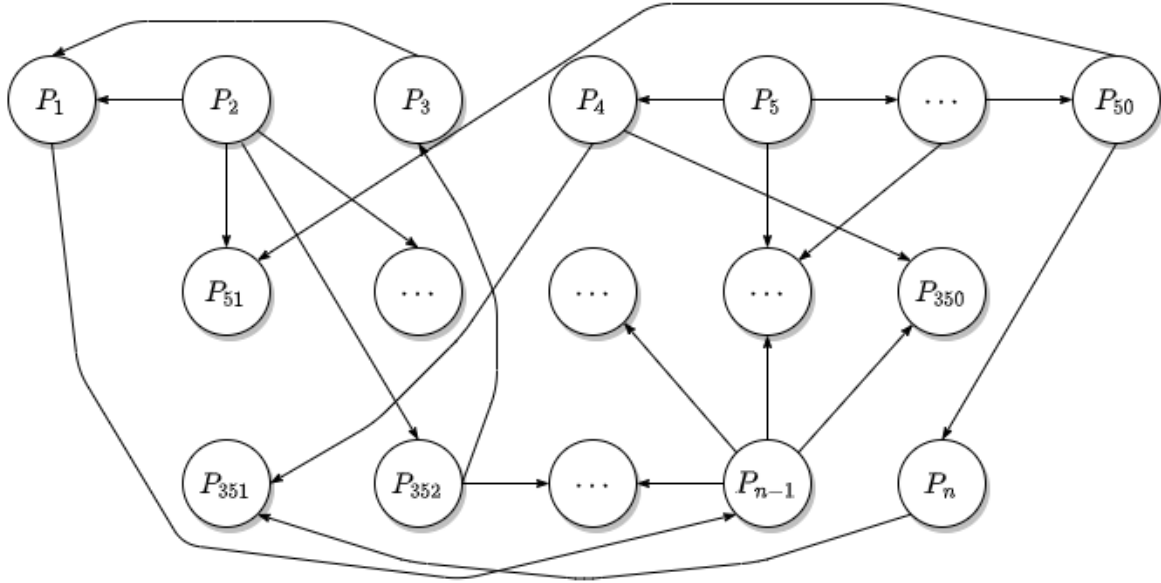
El algoritmo de PageRank utilizado para ordenar las páginas web tiene una base matemática de Álgebra Lineal: transformaremos el problema de ordenar las páginas web según su interés en un problema clásico de valores propios y vectores propios.

En primer lugar, vamos a definir el marco donde nos encontramos. Nuestro problema actual es establecer un criterio para ordenar las páginas de la red. Si llamamos  $P_1, \dots, P_n$  a cada una de las páginas, siendo  $n$  un número natural ( $n \in \mathbb{N}$ ), definimos entonces la importancia  $x_i$  de la página  $P_i$  como un número real entre 0 y 1.

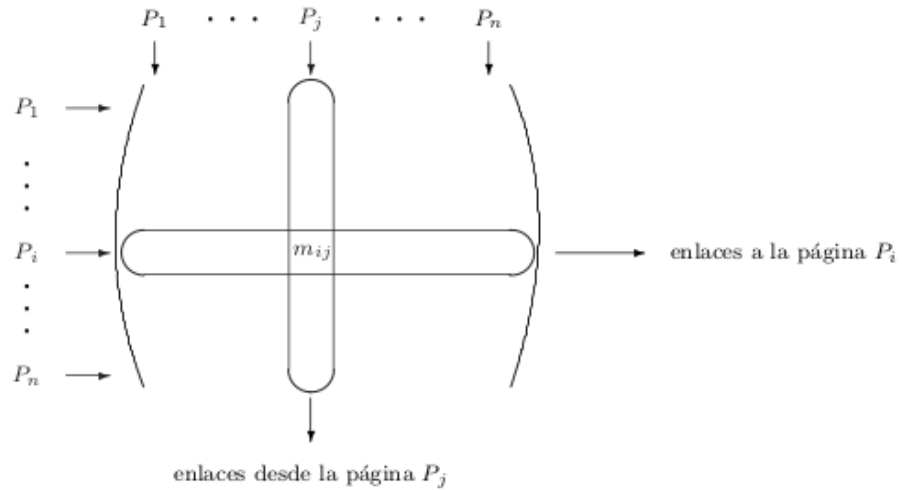
Esta importancia la utilizaremos para elegir qué páginas web son las primeras que mostramos en el buscador, ordenándolas de mayor a menor valor de importancia. Para calcular este número nos basamos en la información que podemos extraer de la red (sitios, contenido, enlaces de una página web a otra, etc).

En el primer modelo nos quedamos solo con los enlaces entre páginas web. Estos enlaces los podemos representar mediante un grafo dirigido ( $G$ ), en el cual representamos cada  $P_i$  como un nodo del grafo y por cada enlace de una página  $P_i$  a  $P_j$  añadimos una arista de  $P_j$  a  $P_i$  indicando el final con una punta de flecha, como se muestra en la imagen.

## 7. Modelo matemático



Esta información la podemos reflejar en una matriz  $M$  de dimensión  $n \times n$ . Tanto en las filas como en las columnas representamos las  $n$  páginas y por cada enlace entre una página  $j$  a otra página  $i$ , escribimos un 1 en la entrada de la matriz  $m_{ij}$  y en el caso de que no haya enlace escribimos un 0.



Por lo tanto en la columna  $j$  estarán los enlaces que salen de la página  $P_j$  hacia otras páginas mientras que en la fila  $i$  estarán representados los enlaces a la página  $P_i$ . Esta matriz no es simétrica ya que una página puede estar citada por otra y ella no citar a ninguna.

En una primera aproximación de querer ordenar las páginas web por “importancia” o “relevancia” podemos pensar que la página a la que le lleguen más enlaces (fila con mayor número de 1) es la que debería mostrarse primero en el *ranking*. Es decir si  $x_j$  es la importancia de  $P_j$ , esta es proporcional al número de páginas desde las que hay enlaces a  $P_j$ .

Sin embargo, la realidad es que el número de enlaces a una cierta página no representa del todo su importancia, ya que no es lo mismo que esta esté citada por una página cualquiera a que esté citada desde [www.facebook.com](http://www.facebook.com) o [www.apple.com](http://www.apple.com). Estos dos últimos enlaces deberían ponderar más en importancia que otras citas de otras páginas web menos relevantes. Por lo que para asignar importancia a una página web, deberemos tener en cuenta tanto si es una **página muy citada** como si es una **página poco citada, pero de sitio “relevantes”**.

Nos damos cuenta de que la importancia de la página citadora también es relevante, por lo que en una segunda aproximación pasamos a decidir que la importancia  $x_j$  de una página  $P_j$  es proporcional a la suma de las importancias de las páginas que enlazan con  $P_j$ . El cambio reside en que en la primera aproximación del modelo la importancia de  $P_i$  era proporcional al número de páginas que enlazan con  $P_i$ , mientras que en esta segunda aproximación la importancia de  $P_i$  es proporcional a la suma de las importancias de las páginas que enlazan con  $P_i$ .

Supongamos, por ejemplo, que la página  $P_1$  es citada desde las páginas  $P_{200}$  y  $P_n$ , que  $P_2$  se cita desde  $P_1$ ,  $P_{50}$ ,  $P_{200}$  y  $P_n$ , mientras que en la última página  $P_n$  hay enlaces desde  $P_1$ ,  $P_2$ ,  $P_{50}$ ,  $P_{200}$  y  $P_{n-1}$ . En nuestra asignación anterior,  $x_1, \dots, x_n$  deberían cumplir entonces que:

$$\begin{aligned} x_1 &= K(x_{200} + x_n) \\ x_2 &= K(x_{50} + x_{200} + x_{n-1}) \\ &\vdots \\ x_n &= K(x_1 + x_2 + x_{50} + x_{200} + x_{n-1}) \end{aligned}$$

donde  $K$  es una constante de proporcionalidad. Si nos fijamos, hemos construido un sistema de ecuaciones donde las soluciones son los posibles valores de  $x_1, \dots, x_n$ . Este sistema de ecuaciones lo podemos escribir en términos matriciales:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = K \begin{pmatrix} \begin{matrix} P_1 & P_2 & & P_{50} & & P_{200} & & P_{n-1} & P_n \\ \downarrow & \downarrow & & \downarrow & & \downarrow & & \downarrow & \downarrow \end{matrix} \\ \begin{matrix} 0 & 0 & \dots & 0 & \dots & \dots & 1 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & \dots & \dots & 1 & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & \dots & \dots & 1 & \dots & 1 & 0 \end{matrix} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Si llamamos  $\vec{x}$  al vector de importacias  $(x_1 \dots x_n)$ ,  $\lambda = \frac{1}{K}$  y  $M$  a la matriz de dimensiones  $n \times n$  del sistema (matriz asociada al grafo). Nos encontramos con un problema de valores propios y vectores propios:

$$M\vec{x} = \lambda\vec{x}$$

A este vector  $\vec{x}$  le exigimos que sea no negativo, es decir, que todas sus componentes sean no negativas, ya que cada una de sus componentes representa la importancia de una página web y también buscamos que sea “único”, en el sentido de que cualquier otro vector propio asociado a ese mismo valor propio sea múltiplo de este, ya que como nuestro objetivo final es establecer un *ranking*, de haber dos vectores linealmente independientes tendríamos dos formas distintas de ordenar y habría que diseñar otro proceso de selección.

El estudio de la existencia de vector propio con esas propiedades para matrices con todas sus entradas no negativas es el objetivo de la Teoría de Perron-Frobenius que desarrollaremos en la siguiente sección.

## 8. Teorema de Perron-Frobenius

Como hemos dicho anteriormente, en esta sección nos centraremos en el Teorema de Perron-Frobenius. Este resultado garantiza la existencia de un valor propio cumpliendo las propiedades que son la base del algoritmo del Pagerank.

La mayoría de los resultados aquí expuestos se pueden encontrar, por ejemplo, en [1].

En primer lugar pasamos a introducir algunas notaciones y definiciones que usaremos posteriormente.

Dadas  $A = (a_{ij}), B = (b_{ij}) \in M_n(\mathbb{C})$  y  $\vec{x} = (x_1, \dots, x_n)^t \in \mathbb{C}^n$ :

1. Diremos que  $A$  es no negativa si  $a_{ij} \geq 0$ ,  $1 \leq i, j \leq n$  y lo notaremos  $A \geq 0$ .
2. Diremos que  $A$  es positiva si  $a_{ij} > 0$ ,  $1 \leq i, j \leq n$  y lo notaremos  $A > 0$ .
3. Diremos que  $B$  es mayor o igual que  $A$  si  $a_{ij} \leq b_{ij}$ ,  $1 \leq i, j \leq n$  y lo notaremos  $A \leq B$ .
4. Diremos que el vector  $\vec{x}$  es no negativo si  $x_i \geq 0$ ,  $1 \leq i \leq n$  y lo notaremos  $\vec{x} \geq 0$ .
5. Análogamente diremos que el vector  $\vec{x}$  es positivo si  $x_i > 0$ ,  $1 \leq i \leq n$  y lo notaremos  $\vec{x} > 0$ .

Denotaremos, como es habitual:

1.  $\sigma(A)$  al espectro de  $A$ , es decir  $\sigma(A) = \{\lambda \in \mathbb{C} : \det(A - \lambda I) = 0\}$
2.  $\rho(A)$  al radio espectral de  $A$ , es decir  $\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}$ .
3.  $m(\lambda)$  a la multiplicidad algebraica del valor propio  $\lambda$ , es decir  $m(\lambda)$  es la multiplicidad de  $\lambda$  como raíz del polinomio característico  $p(\lambda) = \det(A - \lambda I)$ .
4.  $\nu(\lambda)$  a la multiplicidad geométrica del valor propio  $\lambda$ , es decir  $\nu(\lambda)$  es la dimensión del subespacio propio generado por  $\lambda$ .

Además llamaremos par propio a la pareja  $(\lambda, \vec{x})$  donde  $\lambda$  es un valor propio y  $\vec{x}$  un vector propio asociado a  $\lambda$ .



Además de estos elementos básicos del Álgebra Lineal, también utilizaremos las formas canónicas de Jordan. La demostración de este resultado, que se estudia habitualmente en el grado, puede verse, por ejemplo en [2].

**Teorema 8.1 (Forma canónica de Jordan).** Existe una matriz  $P \in M_n(\mathbb{C})$ ,  $\det P \neq 0$  y  $J \in M_n(\mathbb{C})$  diagonal por bloques, es decir,  $J = \text{diag}(J_1, \dots, J_s)$ , tales que

$$A = PJP^{-1}$$

$J$  se denomina “forma canónica” de  $A$  y a cada uno de los bloques  $J_i$ ,  $i = 1, \dots, s$ , “bloques elementales” de Jordan.  $J$  viene determinada por las siguientes propiedades:

1. Cada uno de los bloques elementales,  $J_i$ , está asociado a un mismo valor propio,  $\lambda \in \sigma(A)$ , y es de la forma:

$$J_i = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & 1 & \\ & & \ddots & \ddots \\ & & & \lambda_i & 1 \\ & & & & \lambda_i \end{pmatrix} = \lambda_i I_{p_i} + N_{p_i} \text{ siendo } N_{p_i} = \begin{pmatrix} 0 & 1 & & \\ & 0 & 1 & \\ & & \ddots & \ddots \\ & & & 0 & 1 \\ & & & & 0 \end{pmatrix}$$

2. Cada valor propio  $\lambda$  se repite exactamente  $m(\lambda)$  veces en la diagonal de  $J$ .
3. El número de bloques elementales correspondiente al valor propio  $\lambda$  es  $\nu(\lambda)$ .
4. El número de bloques elementales de dimensión  $l$ , si  $\lambda$  es real, viene dado por

$$2\nu_l(\lambda) - \nu_{l-1}(\lambda), \quad 1 \leq l \leq m(\lambda)$$

tomando  $\nu_0(\lambda) = 0$ , por definición. Donde  $\nu_l(\lambda) := \dim \ker(A - \lambda I_n)^l$

La forma canónica de Jordan de una matriz es única salvo reordenación de los bloques.

También utilizaremos la siguiente propiedad conocida, cuya demostración se puede encontrar en [3]

**Proposición 8.1.** Sea  $A \in M_n(\mathbb{C})$  una matriz cuadrada y  $\rho(A)$  su radio espectral. Entonces:

$$A^m \rightarrow 0, m \rightarrow \infty \Leftrightarrow \rho(A) < 1$$

A continuación presentamos una definición que será clave para encontrar el vector propio que estamos buscando.

**Definición 8.1 (Valor propio dominante).** Sea  $A \in M_n(\mathbb{C})$ , se dice que tiene valor propio dominante si el espectro

$$\sigma(A) = \{\lambda_1, \lambda_2, \dots, \lambda_n\} \quad (r \leq n)$$

cumple:

- $\lambda_1 > 0$ .
- $m(\lambda_1) = 1$ .
- $|\lambda_i| < \lambda_1$  para  $i = 2, \dots, r$ .

En lo sucesivo, si una matriz tiene valor propio dominante lo notaremos como  $\lambda_p$ . El siguiente resultado nos va a resultar fundamental. La demostración ha sido extraída de [3]

**Proposición 8.2.** Sea  $A \in M_n(\mathbb{C})$  una matriz con valor propio dominante  $\lambda_p$ . Entonces la sucesión  $\{\frac{1}{\lambda_p^k} A^k\}_{k \geq 0}$  converge a una matriz  $Q \in M_n(\mathbb{C})$  que cumple:

$$\text{Im}Q = \text{Ker}(A - \lambda_p I), Q^2 = Q, QA = AQ$$

Se dice que  $Q$  es una **proyección espectral de A**.

*Demostración.* Como  $\lambda_p$  es simple la matriz de Jordan asociada quedaría

$$J = \left( \begin{array}{c|ccc} \lambda_p & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & J^* & \\ 0 & & & \end{array} \right)$$

donde  $J^*$  contiene los valores propios restantes  $\lambda_2, \dots, \lambda_r$ . Como  $\lambda_p$  es valor propio dominante tenemos,

$$r(J^*) = \max\{|\lambda_i| : i = 2, \dots, r\} < \lambda_p$$

Esto equivale a  $r(\frac{1}{\lambda_p} J^*) < 1$ , si aplicamos el resultado 8.1 a la matriz  $\frac{1}{\lambda_p} J^*$  obtenemos que  $\frac{1}{\lambda_p^k} (J^*)^k \rightarrow 0$ .

## 8. Teorema de Perron-Frobenius

Si vemos  $J$  como una matriz diagonal por bloques podemos deducir:

$$J^k = \left( \begin{array}{c|ccc} \lambda_p^k & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & (J^*)^k & \\ 0 & & & \end{array} \right)$$

Utilizando la continuidad del producto tenemos

$$\frac{1}{\lambda_p^k} A^k = P \left( \frac{1}{\lambda_p^k} J^k \right) P^{-1} \rightarrow P \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right) P^{-1}$$

Por lo que queda probada la existencia del límite

$$\lim_{k \rightarrow \infty} \frac{1}{\lambda_p^k} A^k = Q \text{ con } Q = P \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right) P^{-1}$$

Comprobamos la propiedades que debe cumplir  $Q$ .

- $ImQ = Ker(A - \lambda_p I)$ .

Si expresamos la matriz de paso por columnas  $P = (p_1 | p_2 | \dots | p_n)$  de la identidad  $AP = PJ$  podemos deducir que  $Ap_1 = \lambda_p p_1$ , ya que:

$$AP = A(p_1 | p_2 | \dots | p_n) = (Ap_1 | Ap_2 | \dots | Ap_n)$$

$$PJ = (p_1 | p_2 | \dots | p_n) \left( \begin{array}{c|ccc} \lambda_p & 0 & \dots & 0 \\ \hline 0 & & & \\ \vdots & & J^* & \\ 0 & & & \end{array} \right) = (\lambda_p p_1 | \star | \dots | \star)$$

Y como  $\det(P) \neq 0$  el vector  $p_1$  no es nulo, de donde deducimos que  $ker(A - \lambda_p I)$  tiene dimensión 1 ya que  $ker(A - \lambda_p I) = \langle \vec{p} \rangle$ .

De la definición de  $Q$  deducimos

$$QP = P \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right)$$

$$QP = Q(p_1|p_2|\dots|p_n) = (Qp_1|Qp_2|\dots|Qp_n)$$

$$P \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right) = (p_1|p_2|\dots|p_n) \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right) = (p_1|0|\dots|0)$$

Por lo que  $Qp_1 = p_1$ ,  $Qp_2 = 0$ ,  $\dots$ ,  $Qp_n = 0$ . Y como  $p_1, \dots, p_n$  es una base de  $\mathbb{C}^n$  deducimos que  $\text{Im}Q = \langle p_1 \rangle$ .

■  $Q^2 = Q$ .

$$Q^2 = P \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right)^2 P^{-1} = P \left( \begin{array}{c|ccc} 1 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array} \right) P^{-1} = Q$$

■  $QA = AQ$

Utilizando la continuidad del producto tenemos que

$$QA = \left( \lim_{k \rightarrow \infty} \frac{1}{\lambda_p^k} A^k \right) A = \lim_{k \rightarrow \infty} \left( \frac{1}{\lambda_p^k} A^k \cdot A \right) = \lim_{k \rightarrow \infty} \left( A \cdot \frac{1}{\lambda_p^k} A^k \right) \stackrel{\text{c.p.}}{=} A \left( \lim_{k \rightarrow \infty} \frac{1}{\lambda_p^k} A^k \right) = AQ$$

□

Tras estos resultados, vemos la importancia de que una matriz  $A$  tenga valor propio dominante, sin embargo, no todas las matrices lo tienen. En la siguiente sección vamos a probar que podemos asegurar la existencia de valor propio dominante cuando la matriz tiene todas sus entradas positivas.

## 8.1. Matrices positivas

Comenzamos esta sección enumerando algunas propiedades elementales de las matrices positivas y no negativas, cuya demostración es un simple ejercicio.

**Proposición 8.3.** Sean  $A \in M_n(\mathbb{C})$  y  $\vec{x}, \vec{y} \in \mathbb{C}^n$ . Entonces se verifica lo siguiente:

$$A > 0, \vec{x} \geq 0, \vec{x} \neq 0 \Rightarrow A\vec{x} > 0 \quad (1)$$

8. Teorema de Perron-Frobenius

$$A \geq 0, \vec{x} \geq \vec{y} \geq 0 \Rightarrow A\vec{x} > A\vec{y} \quad (2)$$

$$A \geq 0, \vec{x} > 0, A\vec{x} = 0 \Rightarrow A = 0 \quad (3)$$

$$A > 0, \vec{y} > \vec{x} > 0 \Rightarrow A\vec{y} > A\vec{x} \quad (4)$$

A continuación veamos unas propiedades básicas de las matrices positivas.

**Proposición 8.4.** Sea  $A \in M_n(\mathbb{C}), A > 0$ , entonces se cumple:

- $\rho(A) \in \sigma(A)$ .
- $\exists \vec{v} \in (A - \rho(A)I), \vec{v} > 0$ .

*Demostración.* Podemos asumir que  $\rho(A) = 1$  sin perder generalidad ya que  $A$  siempre puede ser normalizada por su radio espectral ( $A > 0 \Leftrightarrow \frac{1}{\rho(A)}A > 0$  y  $\rho(A) = r \Leftrightarrow \rho(\frac{1}{r}A) = 1$ ). Si  $(\lambda, \vec{x})$  es un par propio de  $A$  tal que  $|\lambda| = 1$  entonces se cumple que:

$$|\vec{x}| = |\lambda||\vec{x}| = |\lambda\vec{x}| = |A\vec{x}| \leq |A||\vec{x}| = A|\vec{x}| \Rightarrow |\vec{x}| \leq A|\vec{x}|. \quad (5)$$

Donde  $|\vec{x}|$  es el módulo del vector. Por comodidad llamamos  $\vec{z} = A|\vec{x}|$  y llamamos  $\vec{y} = \vec{z} - |\vec{x}|$ , por 5 tenemos que  $\vec{y} \geq 0$ . Supongamos que  $\vec{y} \neq 0$ , es decir que alguna de sus componentes es positiva ( $y_i > 0$ ). En este caso por 1 sabemos que  $A\vec{y} > 0$  y  $\vec{z} > 0$ , por lo tanto debe existir un  $\epsilon > 0$  tal que  $A\vec{y} > \epsilon\vec{z}$  o equivalentemente:

$$\frac{A}{1+\epsilon}\vec{z} > \vec{z}$$

Si escribimos esta inecuación como  $B\vec{z} > \vec{z}$  donde  $B = A/(1+\epsilon)$ , y multiplicamos sucesivamente a ambos lados por  $B$  utilizando 4 tenemos que:

$$B^2\vec{z} > B\vec{z} > \vec{z}, B^3\vec{z} > B^2\vec{z} > \vec{z}, \dots \Rightarrow B^k\vec{z} > \vec{z} \quad \forall k = 1, 2, \dots$$

Pero  $\lim_{k \rightarrow \infty} B^k = \vec{0}$  porque  $\rho(B) = \sigma(A/(1+\epsilon)) = 1/(1+\epsilon) < 1$ , por lo tanto en el límite tenemos que  $\vec{0} > \vec{z}$  lo que contradice el hecho de que  $\vec{z} > 0$ , es decir nuestra suposición de que  $\vec{y} \neq 0$  es incorrecta, por lo que se cumple que  $0 = \vec{y} = A|\vec{x}| - |\vec{x}|$ . Este  $|\vec{x}|$  es un vector propio de  $A$  asociado con el valor propio  $1 = \rho(A)$  y concluimos demostrando que  $|\vec{x}| > 0$  ya que  $|\vec{x}| = A|\vec{x}| = \vec{z} > 0$ .  $\square$

**Proposición 8.5.** Sea  $A \in M_n(\mathbb{C}), A > 0$ . Si  $\lambda \in \sigma(A)$  y  $|\lambda| = \rho(A)$ . Entonces  $\lambda = \rho(A)$ .

## 8. Teorema de Perron-Frobenius

*Demostración.* Supongamos de nuevo que  $\rho(A) = 1$  sin perder generalidad. Sabemos por 8.4 que si  $(\lambda, \vec{x})$  es un par propio de  $A$  tal que  $|\lambda| = 1$ , entonces  $0 < |\vec{x}| = A|\vec{x}|$ , y  $0 < |x_k| = (A|\vec{x}|)_k = \sum_{j=1}^n a_{kj}|x_j|$ . Pero también es cierto que  $|x_k| = |\lambda||x_k| = |(\lambda\vec{x})_k| = |(A\vec{x})_k| = |\sum_{j=1}^n a_{kj}x_j|$ , y entonces se da que

$$\left| \sum_j a_{kj}x_j \right| = \sum_j a_{kj}|x_j| = \sum_j |a_{kj}x_j| \quad (6)$$

Para vectores no negativos  $\{\vec{z}_1, \dots, \vec{z}_n\} \subset \mathbb{C}^n$ , es cierto que  $\|\sum_j \vec{z}_j\|_2 = \sum_j \|\vec{z}_j\|_2$  (igualdad en la desigualdad triangular) si y solo si cada  $\vec{z}_j = \alpha_j \vec{z}_1$  para algun  $\alpha_j$ . En particular, esto se mantiene para escalares, por lo que 6 nos asegura la existencia de un número  $\alpha_j > 0$  tal que

$$a_{kj}x_j = \alpha_j(a_{k1}x_1) \text{ o equivalentemente } x_j = \pi_j x_1 \text{ con } \pi_j = \frac{\alpha_j a_{k1}}{a_{kj}} > 0$$

En otra palabras, si  $|\lambda| = 1$ , entonces  $\vec{x} = x_1 \vec{p}$  donde  $\vec{p} = (1, \pi_2, \dots, \pi_n)^T > \vec{0}$  entonces

$$\lambda \vec{x} = A\vec{x} \Rightarrow \lambda \vec{p} = A\vec{p} = |A\vec{p}| = |\lambda \vec{p}| = |\lambda| \vec{p} = \vec{p} \Rightarrow \lambda = 1$$

□

**Proposición 8.6.** Sea  $A \in M_n(\mathbb{C})$ ,  $A > 0$  y  $r = \rho(A) = \sigma(A)$ . Entonces  $m(r) = \nu(r)$ , es decir, el radio espectral es un valor propio semisimple.

*Demostración.* Como en las demostraciones anteriores, suponemos que  $\rho(A) = 1$ . Además sabemos que es valor propio; supongamos que  $m(1) > \nu(1)$  y lleguemos a una contradicción (recordamos  $m(\lambda) \geq \nu(\lambda)$ , para cualquier  $\lambda \in \sigma(A)$ ). La forma canónica de Jordan de  $A$  tiene entonces un bloque de Jordan asociado al valor propio 1 de orden de al menos dos. Reordenando apropiadamente los bloques de Jordan,

$$A = P \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & J^* & \\ 0 & 0 & & & \end{pmatrix} P^{-1} \Rightarrow A^k = P \begin{pmatrix} 1 & k & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & (J^*)^k & \\ 0 & 0 & & & \end{pmatrix} P^{-1}$$

y  $\|A^k\|_\infty \rightarrow \infty$ , donde,  $\|A\|_\infty = \max_i \sum_j |a_{ij}|$  (como es habitual). Pongamos que  $A^k = (a_{ij}^{(k)})_{i,j=1,\dots,n}$  y sea  $i_k \in \{1, \dots, n\}$  la columna de  $A$  cuya suma sea máxima, es decir, tal

que:

$$\|A^k\|_\infty = \sum_j |a_{ij}^{(k)}| = \sum_j a_{ij}^{(k)} \text{ porque } A \geq 0 \Rightarrow A^k \geq 0 \text{ para cualquier } k$$

Por la propiedad anterior 8.6 sabemos que  $\exists \vec{v} > 0$  tal que  $A\vec{v} = \vec{v}$  ( $\vec{v} \in \text{Ker}(A - I)$ ), recordamos que hemos tomado  $\rho(A) = 1$ ). Entonces  $A^k\vec{v} = \vec{v}$  y

$$\|\vec{v}\|_\infty \geq v_{i_k} = \sum_j a_{ij}^{(k)} v_j \geq \left( \sum_j a_{ij}^{(k)} \right) \min_j v_j = \|A^k\|_\infty \min_j v_j$$

Como  $\min_j v_j > 0$  puesto que  $\vec{v} > 0$ ,  $\|A^k\|_\infty \rightarrow \infty$  llegando a una contradicción ya que  $\|A^k\|_\infty \leq \frac{\|\vec{v}\|_\infty}{\min_j v_j}$  para cualquier  $k \in \mathbb{N}$ .  $\square$

**Proposición 8.7.** Sea  $A \in M_n(\mathbb{C})$ ,  $A > 0$  entonces  $m(\rho(A)) = 1$  y  $\rho(A)$  es el valor propio dominante de  $A$ .

*Demostración.* Supongamos de nuevo que  $\rho(A) = 1$  sin perder generalidad y supongamos también que  $m(\lambda = 1) = m_1 > 1$ . Sabemos de la proposición anterior que  $\lambda = 1$  es un valor semisimple ( $m(\lambda) = \nu(\lambda)$ ), por lo tanto hay  $m_1$  vectores propios linealmente independientes asociados con  $\lambda = 1$ . Sean  $\vec{x}$  e  $\vec{y}$  dos vectores propios asociados con  $\lambda = 1$ , entonces  $\vec{x} \neq \alpha \vec{y}$  para todo  $\alpha \in \mathbb{C}$ . Si elegimos una componente de  $\vec{y}$  no nula, por ejemplo  $y_i \neq 0$ , y definimos  $\vec{z} = \vec{x} - (x_i/y_i)\vec{y}$ , entonces  $A\vec{z} = \vec{z}$ , además de 8.4 sabemos que  $A|\vec{z}| = |\vec{z}| > 0$ . Pero esto contradice el hecho de  $z_i = x_i - (x_i/y_i)y_i = 0$ . Por lo tanto la suposición  $m_1 > 1$  es falsa, concluimos que  $m_1 = 1$ .  $\square$

Dado que  $m(\lambda_p) = 1$ ,  $\nu(\lambda_p) = 1$  denotaremos por  $\vec{p}$  al vector propio asociado a  $\lambda_p$  que cumple  $\|\vec{p}\|_1 = 1 = \sum |p_i| = 1$  y lo llamaremos **vector de Perron**. Donde  $\|\cdot\|_1$  es la norma 1 de un vector y se calcula de la siguiente forma  $\|\vec{x}\|_1 = \sum_{i=1}^n |x_i|$

Otro resultado importante de las matrices positivas, que posteriormente tomará un papel importante en la demostración del Teorema de Perron-Frobenius es el siguiente:

**Proposición 8.8.** Sea  $A \in M_n(\mathbb{C})$ ,  $A > 0$ . Entonces no hay más vectores propios no negativos ( $\vec{x} \geq 0$ ) de  $A$  que no sean el vector de Perron y sus múltiplos positivos.

## 8. Teorema de Perron-Frobenius

*Demostración.* Llamamos  $(\lambda, \vec{y})$  al par propio de  $A$  tal que  $\vec{y} \geq 0$  y llamamos  $\vec{p}_t > 0$  al vector de Perron para  $A^T$ . Como  $\vec{y} \neq \vec{0}$  por ser vector propio, entonces por 1 tenemos que  $\langle \vec{p}_t, \vec{y} \rangle > 0$ . Por lo tanto:

$$\begin{aligned} \rho(A)\vec{p}_t = A^T \vec{p}_t &\Rightarrow \langle \rho(A)\vec{p}_t, \vec{y} \rangle = \langle A^T \vec{p}_t, \vec{y} \rangle = \langle \vec{p}_t, A\vec{y} \rangle = \langle \vec{p}_t, \lambda \vec{y} \rangle = \langle \lambda \vec{p}_t, \vec{y} \rangle \Rightarrow \\ \rho(A) &= \lambda \end{aligned}$$

□

Todas estas propiedades sobre las matrices positivas las podemos resumir en el teorema de Perron.

**Teorema 8.2 (Perron, 1907).** Sea  $A \in M_n(\mathbb{C})$ ,  $A > 0$  con  $\lambda_p = \rho(A)$ . Entonces:

1.  $\lambda_p > 0$ .
2.  $\lambda_p \in \sigma(A)$  ( $\lambda_p$  es llamado raíz de Perron).
3.  $m(\lambda_p) = 1$ .
4. Existe un vector propio  $\vec{v} > 0$  tal que  $A\vec{v} = \lambda_p \vec{v}$ .
5. El **vector de Perron** es el único definido como

$$A\vec{p} = \lambda_p \vec{p}, \vec{p} > 0, \text{ y } \|\vec{p}\|_1 = 1$$

y, excepto los múltiplos positivos de  $\vec{p}$ , no hay otros vectores propios no negativos para  $A$ , independientemente del valor propio.

6.  $\lambda_p$  es el único valor propio en el círculo espectral de  $A$ .

Por lo tanto, toda matriz positiva tiene un valor propio dominante cuyo vector propio asociado cumple las propiedades que estamos buscando. Sin embargo, nuestra matriz no es positiva ya que varias de sus entradas son ceros.

En la siguiente sección vemos como se adapta del Teorema de Perron a las matrices no negativas, surgiendo así el Teorema de Perron-Frobenius.



## 8.2. Matrices no negativas

Como hemos dicho antes en esta sección vemos como se adaptan los resultados anteriores a las matrices no negativas. Comenzamos con unas definiciones y propiedades básicas sobre las matrices no negativas.

**Proposición 8.9.** Sea  $A \in M_n(\mathbb{C})$ ,  $A \geq 0$  con  $r = \rho(A)$ , entonces se cumple:

$$r \in \sigma(A) \quad (r = 0 \text{ es posible}) \quad (7)$$

$$A\vec{z} = r\vec{z} \text{ para algún } \vec{z} \in N = \{\vec{x} | \vec{x} \geq 0 \text{ con } \vec{x} \neq 0\} \quad (8)$$

*Demostración.* Consideramos la secuencia de matrices positivas  $A_k = A + (1/k)E > 0$ , donde  $E$  es la matriz cuyas entradas son todo 1, y sean  $\lambda_{p_k} > 0$  y  $\vec{p}_k > 0$  al valor propio dominante y al vector de Perron de  $A_k$  respectivamente. Observamos que  $\{\vec{p}_k\}_{k=1}^{\infty}$  es un conjunto acotado ya está contenido en la esfera unidad de  $\mathbb{R}^n$ . El teorema de Bolzano-Weierstrass que esa sucesión admite una subsucesión parcial convergente.

Por lo tanto  $\{\vec{p}_{k_i}\}_{i=1}^{\infty} \rightarrow \vec{z}$ , donde  $\vec{z} \geq 0$  con  $\vec{z} \neq 0$  (ya que  $p_{k_i} > 0$  y  $\|p_{k_i}\|_1 = 1$ ). Como  $A_1 > A_2 > \dots > A$  se cumple que  $\lambda_{p_1} \geq \lambda_{p_2} \geq \dots \geq r$ , por lo tanto  $\{\lambda_{p_k}\}_{k=1}^{\infty}$  es una sucesión monótona de números positivos acotada inferiormente por  $r$ . Un resultado estandar del análisis garantiza que

$$\lim_{k \rightarrow \infty} \lambda_{p_k} = \lambda_p^* \text{ existe, y } \lambda_p^* \geq r. \text{ En particular, } \lim_{i \rightarrow \infty} \lambda_{p_{k_i}} = \lambda_p^* \geq r$$

Pero  $\lim_{k \rightarrow \infty} A_k = A$  implica que  $\lim_{i \rightarrow \infty} A_{k_i} \rightarrow A$  así que utilizando que el límite del producto es el producto de los límites (asegurando previamente que todos los límites existen), se tiene que:

$$A\vec{z} = \lim_{i \rightarrow \infty} A_{k_i} \vec{p}_{k_i} = \lim_{i \rightarrow \infty} \lambda_{p_{k_i}} \vec{p}_{k_i} = \lambda_p^* \vec{z} \Rightarrow \lambda_p^* \in \sigma(A) \Rightarrow \lambda_p^* \leq r$$

Concluimos  $\lambda_p^* = r$ , y  $A\vec{z} = r\vec{z}$  con  $\vec{z} \geq 0$  y  $\vec{z} \neq 0$ . □

**Definición 8.2 (Matriz reducible).** Se dice que una matriz  $A \in M_n(\mathbb{R})$  no negativa ( $A \geq 0$ ) es reducible si existe una permutación simétrica (de filas y columnas) que transforma  $A$  en una matriz del tipo:

$$\left( \begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array} \right)$$

donde  $A_{11}$  y  $A_{22}$  son matrices cuadradas. Una matriz se dice irreducible cuando no es reducible. Se demuestra ver [1].

**Proposición 8.10.** Sea  $A \in M_n(\mathbb{R})$  una matriz no negativa ( $A \geq 0$ ). Son equivalentes:

1.  $A$  es irreducible.
2. La matriz  $(I + A)^{n-1}$  es positiva.
3. Si  $A$  es la matriz de adyacencia de un grafo, entonces el grafo está fuertemente conectado.

**Definición 8.3 (Grafo fuertemente conectado).** Un grafo dirigido  $G$  se dice fuertemente conectado si para cada par de nodos distintos  $P_i, P_j$  en  $G$  hay un camino de longitud finita que comienza en  $P_i$  y termina en  $P_j$ .

Con las últimas equivalencias podemos ver como relacionar las propiedades de un grafo con las de su matriz de adyacencia. Algo realmente interesante para nuestro problema, ya que la información que originalmente tenemos proviene de un grafo. Por último, antes de pasar a demostrar el teorema de Perron-Frobenius veamos un resultado que nos ayudará a simplificar su demostración.

**Proposición 8.11.** Sea  $A \in M_n(\mathbb{C})$  y  $\lambda \in \sigma(A)$ . Dado  $k \in \mathbb{N}$  fijo, ponemos  $B = (A + I)^k$  y  $\mu = (1 + \lambda)^k$ . Entonces,  $\mu \in \sigma(B)$  y  $\ker(A - \lambda I) \subseteq \ker(B - \mu I)$ . Además,  $\ker(A - \lambda I)^2 \subseteq \ker(B - \mu I)^2$ .

*Demostración.* La demostración se basa en que, como  $A$  y  $I$  conmutan

$$B = (A + I)^k = \sum_{j=0}^k \binom{k}{j} A^j$$

Sea  $\vec{v} \in \ker(A - \lambda I) \Leftrightarrow A\vec{v} = \lambda\vec{v}$  y, por consiguiente,  $A^j\vec{v} = \lambda^j\vec{v}$ ,  $\forall j \in \mathbb{N}$ . Vamos a demostrar que  $\vec{v} \in \ker(B - \mu I)$ . En efecto

$$B\vec{v} = \left( \sum_{j=0}^k \binom{k}{j} A^j \right) \vec{v} = \sum_{j=0}^k \binom{k}{j} A^j \vec{v} = \sum_{j=0}^k \binom{k}{j} \lambda^j \vec{v} = \left( \sum_{j=0}^k \binom{k}{j} \lambda^j \right) \vec{v} = (1 + \lambda)^k \vec{v} = \mu \vec{v}$$

entonces  $\mu \in \sigma(B)$  y  $\ker(A - \lambda I) \subseteq \ker(B - \mu I)$ . Veamos ahora que  $\ker(A - \lambda I)^2 \subseteq \ker(B - \mu I)^2$ . Sea  $\vec{w} \in \ker(A - \lambda I)^2$ . Como  $\ker(A - \lambda I) \subseteq \ker(B - \mu I) \subseteq \ker(B - \mu I)^2$

## 8. Teorema de Perron-Frobenius

solo tenemos que probarlo para  $\vec{w} \in \ker(A - \lambda I)^2 \setminus \ker(A - \lambda I)$ . Si tal  $\vec{w}$  existe, entonces llamando  $\vec{v} = (A - \lambda I)\vec{w}$ , tenemos que  $\vec{v} \neq 0$  y  $\vec{v} \in \ker(A - \lambda I)$ . Ahora bien,  $\vec{v} = (A - \lambda I)\vec{w}$  implica que  $A\vec{w} = \vec{v} + \lambda\vec{w}$ , por tanto

$$A^2\vec{w} = A\vec{v} + \lambda A\vec{w} = \lambda\vec{v} + \lambda\vec{v} + \lambda^2\vec{w} = 2\lambda\vec{v} + \lambda^2\vec{w}$$

Por recurrencia,

$$A^j\vec{w} = j\lambda^{j-1}\vec{v} + \lambda^j\vec{w}$$

Entonces,

$$\begin{aligned} B\vec{w} &= \left( \sum_{j=0}^k \binom{k}{j} A^j \right) \vec{w} = \sum_{j=0}^k \binom{k}{j} (j\lambda^{j-1}\vec{v} + \lambda^j\vec{w}) = \\ &= \left( \sum_{j=0}^k \binom{k}{j} \lambda^j \right) \vec{w} + \left( \sum_{j=1}^k \binom{k}{j} j\lambda^{j-1} \right) \vec{v} = \mu\vec{w} + a\vec{v} \end{aligned}$$

con  $a \in \mathbb{C}$ , luego  $(B - \mu I)\vec{w} = a\vec{v} \in \ker(A - \lambda I) \subseteq \ker(B - \mu I) \Rightarrow (B - \mu I)^2\vec{w} = 0$ ; esto es,  $\vec{w} \in \ker(B - \mu I)^2$ . □

### 8.3. Enunciado del teorema

Unos años después Frobenius publica un resultado similar para matrices no negativas e irreducibles.

**Teorema 8.3 (Frobenius, 1908-1912).** Sea  $A$  una matriz cuadrada ( $A \in M_n(\mathbb{C})$ ) no negativa ( $A \geq 0$ ). Si  $A$  es irreducible, entonces para  $r = \rho(A)$  se cumple que:

1.  $r > 0$ ,  $r \in \sigma(A)$  y
2.  $m(r) = 1$ .
3. Existe un vector propio  $\vec{x} > 0$  asociado a  $r$ .
4. El único vector definido por:

$$A\vec{p} = r\vec{p}, \vec{p} > 0, \text{ y } \|\vec{p}\|_1 = 1$$

es el vector de Perron. No hay ningún vector propio no negativo para  $A$  excepto los múltiplos positivos de  $\vec{p}$ , independientemente del valor propio.

Esencialmente el teorema de Perron-Frobenius nos asegura que toda matriz no negativa e irreducible siempre tiene un valor propio que es mayor o igual en módulo que todos los demás valores propios y que lleva asociado un vector propio positivo.

### 8.4. Demostración del teorema

*Demostración.* Por la proposición anterior 8.11 tenemos que  $\mu = (1 + r)^{n-1} \in \sigma(B)$  y  $\vec{v} \in \ker(B - \mu I)$ . Además como  $A$  es irreducible, entonces  $(A + I)^{n-1} = B > 0$ . Por el teorema de Perron 8.2 necesariamente  $\mu = \rho(B)$ . Aplicando la propiedad 1, como  $B > 0$ ,  $\vec{v} \geq 0$ , entonces  $\mu\vec{v} = B\vec{v} > 0$  y como  $\mu > 0$  tenemos  $\vec{v} > 0$ .

Sea  $\vec{x} \geq 0$  un vector propio asociado a  $r$  para  $A$ , que existe por la proposición 8.9, entonces  $\vec{x}$  es vector propio asociado a  $\mu$  para la matriz  $B$  ya que por la proposición anterior  $\ker(A - rI) \subseteq \ker(B - \mu I)$ . Por lo tanto  $\vec{x} > 0$  por lo demostrado anteriormente.

Por reducción al absurdo  $r$  es positivo, pues si  $r = 0$  entonces  $0 = r\vec{x} = A\vec{x}$ , lo cual es una contradicción pues  $A \geq 0$  y  $\vec{x} > 0$  implica que  $A\vec{x} > 0$ .

Por otro lado  $\mu$  es un valor propio dominante para  $B$ . En particular  $m_B(\mu) = \nu_B(1) = 1$  (multiplicidades algebraica y geométrica de  $\mu$  como valor propio de  $B$ ). Pero por la proposición anterior,  $\ker(A - rI) \subseteq \ker(B - \mu I)$ , luego  $\nu_A(r) = 1$  y se tiene que

$$1 \leq m_A(r) \leq \nu_A(r) = 1 \Rightarrow r \text{ es simple}$$

Veamos por último que no hay ningún vector no negativo para  $A$  excepto los múltiplos positivos de  $\vec{p}$ . Llamamos  $(\lambda, \vec{y})$  al par propio de  $A$  tal que  $\vec{y} \geq 0$  y llamamos  $\vec{p}_t > 0$  al vector de Perron para  $A^T$ . Como  $\vec{y} \neq \vec{0}$  por ser vector propio, entonces por 1 tenemos que  $\langle \vec{p}_t, \vec{y} \rangle > 0$ . Por lo tanto:

$$\begin{aligned} \rho(A)\vec{p}_t = A^T \vec{p}_t &\Rightarrow \langle \rho(A)\vec{p}_t, \vec{y} \rangle = \langle A^T \vec{p}_t, \vec{y} \rangle = \langle \vec{p}_t, A\vec{y} \rangle = \langle \vec{p}_t, \lambda \vec{y} \rangle = \langle \lambda \vec{p}_t, \vec{y} \rangle \Rightarrow \\ \rho(A) &= \lambda \end{aligned}$$

□

El teorema anterior nos garantiza que el radio espectral es positivo y simple. Sin embargo, para poder afirmar que es dominante, éste debe ser mayor estricto en módulo que los demás valores propios. En el teorema anterior nada nos asegura que sea único, de hecho podría existir otro valor propio en el círculo espectral cuyo módulo sea igual al radio espectral. De hecho en [1] se demuestra la siguiente proposición.

**Proposición 8.12.** Si  $A \geq 0$  es irreducible y tiene  $k$  valores propios  $\{\lambda_1, \lambda_2, \dots, \lambda_k\}$  en su círculo espectral, entonces se cumple:

- $m(\lambda_i) = 1$  para  $i = 1, 2, \dots, k$ .
- $\{\lambda_1, \lambda_2, \dots, \lambda_k\}$  son las raíces complejas de  $r^k$ .

$$\{r, rw, rw^2, \dots, rw^{k-1}\}, \text{ donde } w = e^{2\pi i/k}$$

Por lo tanto para garantizar que el radio espectral sea el valor propio dominante debemos exigirle a la matriz una condición adicional.

**Definición 8.4 (Matriz primitiva).** Una matriz  $A$  no negativa e irreducible se dice que es una matriz primitiva si tiene un solo valor propio,  $r = \rho(A)$ , en su círculo espectral.

**Definición 8.5 (Matriz imprimitiva).** Una matriz no negativa e irreducible se dice que es imprimitiva si tiene  $h > 1$  valores propios en su círculo espectral y  $h$  se conoce como el índice de imprimitividad.

Veamos a continuación dos proposiciones que son una herramienta para saber cuando una matriz es primitiva. Se demuestran ver [1].

**Proposición 8.13.** Una matriz  $A \geq 0$  irreducible es primitiva si y solo si existe  $m \in \mathbb{N}$  tal que  $A^m > 0$

**Proposición 8.14.** Si una matriz  $A \geq 0$  irreducible tiene al menos un elemento positivo en la diagonal, entonces  $A$  es primitiva.

Por lo tanto si nuestra matriz  $A \geq 0$  es irreducible y primitiva, podemos asegurar que el radio espectral es dominante. Por otro lado, el teorema anterior nos garantiza la existencia del vector de Perron de la matriz de nuestro problema inicial, pero ¿cómo hallamos este vector cuando nuestra matriz es de grandes dimensiones?. En la siguiente sección se explicará un método para poder aproximar este vector.

## 8.5. Método de las potencias

El **método de las potencias** es el método que implementaremos posteriormente para aproximarnos al vector de Perron, se trata de ir multiplicando la matriz original por un vector inicial iterativamente hasta conseguir una aproximación de un vector asociado al valor propio dominante. Explicamos a continuación su funcionamiento.

Sabemos que si  $A \geq 0$  tiene valor propio dominante,  $A^T$  también ( $A^T \geq 0$ ) y ambos coinciden ( $\sigma(A) = \sigma(A^T)$ ).

Llamamos  $\vec{p}_t$  al vector de Perron de  $A^T$ . Dado  $\vec{v} \in \mathbb{R}^n$ ,  $\vec{v} \geq 0$  no nulo y como  $A^T \vec{p}_t = \lambda_p \vec{p}_t$  tenemos que:

$$0 < \langle \vec{p}_t, \vec{v} \rangle = \frac{1}{\lambda_p} \langle A^T \vec{p}_t, \vec{v} \rangle = \dots = \frac{1}{\lambda_p^k} \langle (A^T)^k \vec{p}_t, \vec{v} \rangle = \frac{1}{\lambda_p^k} \langle \vec{p}_t, (A)^k \vec{v} \rangle \xrightarrow{8.2} \langle \vec{p}_t, c(\vec{v}) \vec{p} \rangle$$

Puesto que la proposición 8.2 nos asegura que  $\frac{1}{\lambda_p^k} A^k \vec{v}$  converge a un múltiplo de  $\vec{p}$ , que denotamos como  $c(\vec{v}) \vec{p}$ , porque depende de  $\vec{v}$ .

Por lo tanto,  $\langle \vec{p}_t, \vec{v} \rangle = c(\vec{v}) \langle \vec{p}_t, \vec{p} \rangle > 0$ , es decir  $c(\vec{v}) > 0$  cualquiera que sea  $\vec{v} \geq 0$

El método de las potencias comienza con un vector inicial  $\vec{x}^{(0)} \in \mathbb{R}^n$ ,  $\vec{x}^{(0)} \geq 0$ , para calcular los siguiente  $\vec{x}^{(k)}$  se utiliza la siguiente fórmula recurrente:

$\vec{x}^{(k+1)} = A \vec{x}^{(k)}$  con  $k \in \mathbb{N}$ , esta fórmula la podemos desarrollar obteniendo  $\vec{x}^{(k)} = A^k \vec{x}^{(0)}$

Por la proposición 8.2 sabemos que:

$$\frac{1}{\lambda_p^k} \vec{x}^{(k)} \rightarrow c(\vec{x}^{(0)}) \vec{p}, c(\vec{x}^{(0)}) > 0 \text{ porque } \vec{x}^{(0)} \geq 0$$

Pongamos que

$$\lambda_k = \frac{\|\vec{x}^{(k+1)}\|_1}{\|\vec{x}^{(k)}\|_1} \text{ y } \lambda_k = \lambda_p \frac{\left\| \frac{1}{\lambda_p^{k+1}} \vec{x}^{(k+1)} \right\|_1}{\left\| \frac{1}{\lambda_p^k} \vec{x}^{(k)} \right\|_1}$$

O bien

$$\left\| \frac{1}{\lambda_p^k} \vec{x}^{(k)} \right\|_1 \cdot \lambda_k = \lambda_p \left\| \frac{1}{\lambda_p^{k+1}} \vec{x}^{(k+1)} \right\|_1$$

Tomando límites en ambos lados y teniendo en cuenta que  $\|\cdot\|$  es continua:

$$c(\vec{x}^{(0)}) \lim_k \lambda_k = \lambda_p c(\vec{x}^{(0)}) \text{ como } c(\vec{x}^{(0)}) \neq 0 \Rightarrow \lambda_p = \lim_k \lambda_k$$

## 8. Teorema de Perron-Frobenius

Si consideramos ahora:

$$\frac{1}{\|\vec{x}^{(k)}\|_1} \vec{x}^{(k)} = \frac{1}{\lambda_p^k} A^k \vec{x}^{(0)} \frac{\lambda_p^k}{\|\vec{x}^{(k)}\|_1}$$

Tomando de nuevo límite en ambos lados y teniendo en cuenta que  $\|\cdot\|$  es continua tenemos:

$$\lim_{k \rightarrow \infty} \frac{1}{\|\vec{x}^{(k)}\|_1} \vec{x}^{(k)} = \lim_{k \rightarrow \infty} \frac{1}{\lambda_p^k} A^k \vec{x}^{(0)} \frac{\lambda_p^k}{\|\vec{x}^{(k)}\|_1} = c(\vec{x}^{(0)}) \vec{p} \frac{1}{c(\vec{x}^{(0)}) \|\vec{p}\|_1}$$

Como  $c(\vec{x}^{(0)}) \neq 0$  y  $\|\vec{p}\|_1 = 1$  concluimos que:

$$\lim_{k \rightarrow \infty} \frac{1}{\|\vec{x}^{(k)}\|_1} \vec{x}^{(k)} = \vec{p}$$

Como acabamos de demostrar, con este método conseguimos un vector  $\vec{x}^{(k)}$  que se aproxima tanto como queramos (aumentando las iteraciones) a un vector proporcional al vector de Perron.



# Parte II.

## Informática

### 9. Esquema del proyecto

En esta sección explicaremos el esquema del proyecto realizado. El objetivo de este proyecto es implementar un buscador capaz de ordenar los archivos mediante el algoritmo de PageRank, filtrar estos archivos mediante una consulta y mostrar un resultado más personalizado dependiendo del usuario que realice la consulta. Se ha elegido como lenguaje de programación *python* por su sencillez y por que cuenta con librerías como *numpy* con muchas herramientas para trabajar con matrices.

Debido a la limitación de los recursos utilizados, para realizar la primera parte del proyecto se ha realizado una selección de 3000 documentos relacionados entre ellos los cuales constan de unas 26000 palabras. Esta división ha sido necesaria puesto que en el intento de implementar el algoritmo con todos los artículos la matriz creada alcanzaba una dimensión muy grande impidiendo que el ordenador utilizado pudiera operar con ella.

En la segunda parte del proyecto utilizamos la teoría de la recuperación de información para implementar un buscador capaz de reordenar los archivos mediante una consulta.

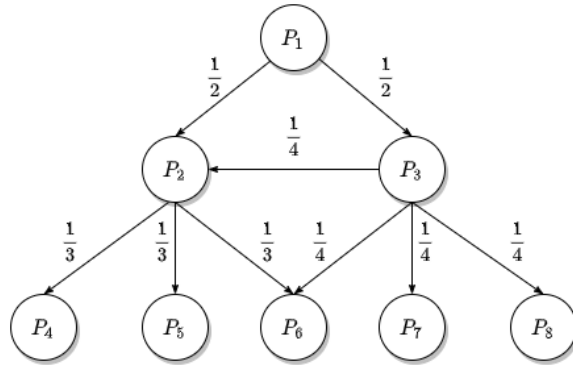
En esta segunda parte inicialmente se implementa un modelo binario donde se eligen aquellos documentos que contienen alguna de las palabras introducidas en la consulta. No se tiene en cuenta su frecuencia en el documento ni su importancia. En un segundo intento se implementa un modelo vectorial, donde se consiguen solventar los problemas del primer modelo. Por último, se implementa una expansión de consulta basada en la relevancia que le otorga el usuario a cada documento. En este último caso los usuarios serán los propios autores ya que no poseemos más información para elaborar algún perfil. Por lo tanto, para cada usuario o autor serán relevantes los documentos que ha publicado.

Comenzamos retomando el modelo matemático del PageRank y tras su explicación, pasaremos a explicar los métodos de Recuperación de Información.

## 10. Modificaciones previas

En primer lugar, vamos a enfocar el problema desde otro punto de vista. Supongamos que el usuario se encuentra navegando por la red, por ejemplo, en la primera página  $P_1$  y cuando se aburre decide saltar a una de las páginas con las que enlaza  $P_1$ . Supongamos que hay  $N_1$  páginas que enlazan desde  $P_1$ , es lógico pensar que todas tienen una probabilidad de ser elegidas del  $\frac{1}{N_1}$  es decir que siguen una distribución de probabilidad uniforme (discreta) en  $[1, N_1]$ .

Construimos entonces un modelo probabilístico, es decir no sabemos que camino elegirá el usuario pero si sabemos con que probabilidad estará en cada uno de los destinos posibles. Por ejemplo, en el siguiente grafo, si el usuario se encuentra en  $P_1$  puede elegir entre dos páginas  $P_2$  y  $P_3$ . Por lo que el usuario tiene una probabilidad de  $\frac{1}{2}$  de elegir cada una de ellas. Si por ejemplo, finalmente elige la página  $P_3$ , entonces volvería a sortear su elección pero esta vez cada página de destino tiene una probabilidad de  $\frac{1}{4}$  de ser elegida.

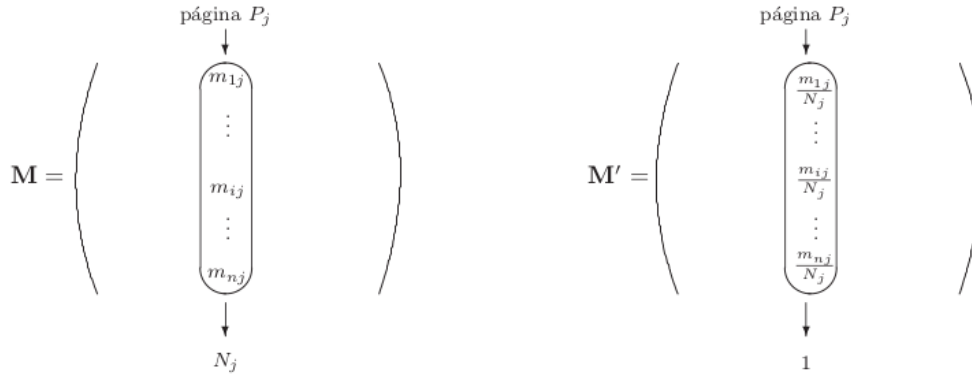


Volvemos a considerar las mismas paginas web  $P_1, P_2, \dots, P_n$  y  $M$  la matriz de adyacencia del grafo, cuyas entradas  $m_{ij}$  son 0 y 1. Llamamos  $N_j$  al número de enlaces de la página  $P_j$ , es decir al número de entradas de la columna  $j$ . Construimos una nueva matriz  $M'$  a partir de la  $M$  original sustituyendo cada  $m_{ij}$  por

$$m'_{ij} = \frac{m_{ij}}{N_j}$$

La transformación sería la siguiente:

## 10. Modificaciones previas



La nueva matriz  $M'$  esta compuesta por números no negativos entre 0 y 1, a esta matriz así construida se le llama una matriz estocástica (o de Markov).

Con este modelo podemos conocer con que probabilidad estará el usuario en cada una de las páginas tras cada instante de tiempo. Por ejemplo, si el usuario parte de la página  $k$ -ésima  $P_k$  para saber que probabilidad tiene de cambiar a cada uno de los posibles destinos solo tenemos que multiplicar la matriz  $M'$  por el vector inicial. En este caso el vector inicial es un vector de ceros y un 1 en la posición  $k$ -ésima, ya que sabemos que el usuario se encuentra en la página  $P_k$  con una probabilidad del 100 %.

Si multiplicamos  $M'$  por el vector inicial, obtenemos:

$$\begin{pmatrix} \cdots & \cdots & m'_{1k} & \cdots \\ \vdots & \ddots & \vdots & \vdots \\ \cdots & & m'_{kk} & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ \cdots & \cdots & m'_{nk} & \cdots \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} m'_{1k} \\ \vdots \\ m'_{kk} \\ \vdots \\ m'_{nk} \end{pmatrix}$$

Este vector resultante, cuyas entradas son 0 o  $1/N_k$ , describe con que probabilidad estará el usuario en cada una de las páginas tras una unidad de tiempo. Para saber con que probabilidad estará en cada uno de los posibles destinos dentro de dos unidades de tiempo solo debemos multiplicar el vector inicial por  $(M')^2$ , si lo queremos saber para dentro de tres unidades de tiempo lo multiplicaremos por  $(M')^3$  y así sucesivamente.

La matriz  $M'$  recibe el nombre de matriz de transición del sistema, ya que en cada entrada  $m'_{ij}$  se refleja la probabilidad de pasar de la página  $P_j$  a la página  $P_i$ . Y además las matrices que se corresponden con sus respectivas potencias también reflejan la probabilidad de pasar de  $P_j$  a  $P_i$  tras varios instantes de tiempo.

Sin embargo, podría ocurrir que alguna de las páginas no citaran a ninguna otra, es decir, que ese nodo del grafo no tuviera enlaces salientes. Esto se traduce en que en nuestra matriz  $M'$  aparece una columna de ceros, por lo que esta matriz dejaría de ser estocástica y además el grafo del que partimos no estaría fuertemente conectado. Para solucionarlo se añade una probabilidad de transición a todos los vértices, es decir, añadimos la posibilidad de que el usuario se “aburra” y decida cambiar a otra página que no esté enlazada con la página en la que se encontraba. En términos matriciales se traduce en lo siguiente:

$$M'' = cM' + (1 - c) \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} (1, \dots, 1)$$

Donde  $p_1, \dots, p_n$  es una distribución de probabilidad y  $c$  es un parámetro entre 0 y 1, el cual Google estima que se encuentra sobre 0,85<sup>4</sup>. La distribución de probabilidad a tomar puede ser una distribución uniforme que asigne una probabilidad de  $p_i = 1/n$ <sup>5</sup> a cada página, aunque también se podría elegir otras, ponderando unas páginas más que otras consiguiendo así búsquedas más personalizadas.

---

<sup>4</sup>Valor extraído de <https://en.wikipedia.org/wiki/PageRank>

<sup>5</sup>Donde  $n$  es el número de Páginas web o documentos.

## 11. Pseudocódigo

En esta sección comentaremos la estructura del algoritmo. Para implementarlo hemos separado el algoritmo en dos funciones. La primera función (**pagerank**) se encarga de construir la matriz de adyacencia del grafo formado por todos los archivos y sus citas, además normaliza esta matriz por columnas para conseguir la matriz de transición del sistema y por último se le añade el parámetro  $c$  y la distribución de probabilidad uniforme como se comenta anteriormente, consiguiendo la matriz  $M''$ . La segunda función (**metodopotencias**) tiene como objetivo calcular el vector propio de nuestra matriz  $M''$ , para ello se utiliza el método de las potencias.

**Data:** Vector que contiene todos los nodos ( $v$ )

**Result:** Vector de pesos del PageRank ( $pg$ )

```

1 m = construyeMatrizAdyaencia(v);
2 for  $i \in \text{len}(v)$  do
3   if  $\text{sum}(m.T[i]) \neq 0$  then
4      $m.T[i] = m.T[i] / \text{sum}(m.T[i])$ ;
5 N = m[0].size();
6 m_seg = (C * m + (1 - C)/N);
7 pg = metodopotencias(m_seg, 100);
8 return pg
```

ALGORITMO 1: pagerank

**Data:**  $M''$  ( $m$ ); Número de interacciones ( $\text{num\_iter}$ )

**Result:** Vector de pesos del PageRank ( $v$ )

```

1 N = m[0].size();
2 v = random(N, 1);
3 v = v / max(v);
4 for  $i \in \text{num\_iter}$  do
5    $v = m \cdot v$ 
6 return v
```

ALGORITMO 2: metodopotencias

En el primer pseudocódigo  $m.T$  se refiere a la traspuesta de  $m$  y la constante  $C$  tiene un valor de 0.85 fijado previamente.

## 12. Ejemplos

Para ilustrar como funciona el algoritmo del PageRank vamos a mostrar unos ejemplos. Comenzamos con el primer ejemplo, en este nos damos cuenta de que tanto  $P_4$  como  $P_5$  tienen dos enlaces entrantes, sin embargo, queda primero en la clasificación  $P_5$ , después  $P_1$  y en tercer lugar  $P_4$ . Esto se debe a que una de los enlaces entrantes de  $P_5$  es de  $P_4$  una de las páginas con más enlaces. Recordemos que este algoritmo premia también si la página que cita es importante.

### Ejemplo 1

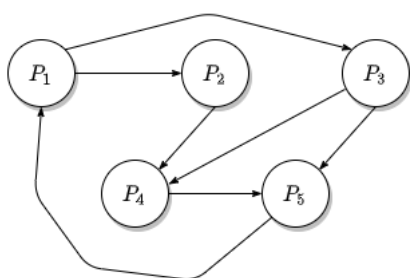


FIGURA 2: Grafo

	Página	PageRank
1º	$P_5$	0.26375504
2º	$P_1$	0.25419178
3º	$P_4$	0.20599017
4º	$P_2$	0.13803151
5º	$P_3$	0.13803151

TABLA 1: Clasificación

Añadimos al ejemplo dos páginas nuevas que también citan a  $P_4$ , sin embargo, esto no hace que  $P_4$  se coloque en primera posición. Queda reflejado que el algoritmo da más puntuación a que la página tenga un enlace de una página importante que a una página con muchos enlaces.

### Ejemplo 2

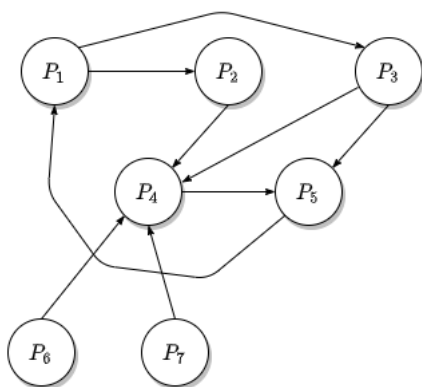


FIGURA 3: Grafo

	Página	PageRank
1º	$P_5$	0.2564552
2º	$P_1$	0.2394155
3º	$P_4$	0.21491184
4º	$P_2$	0.12318016
5º	$P_3$	0.12318016
6º	$P_7$	0.02142857
7º	$P_6$	0.02142857

TABLA 2: Clasificación

Cambiamos la estrategia, en vez de añadir citas desde páginas nuevas hasta  $P_4$  le añadimos una cita desde una de las páginas del grafo inicial,  $P_1$ , y observamos que tampoco conseguimos quitar a  $P_5$  del primer puesto aunque la  $P_4$  tenga más enlaces entrantes.

### Ejemplo 3

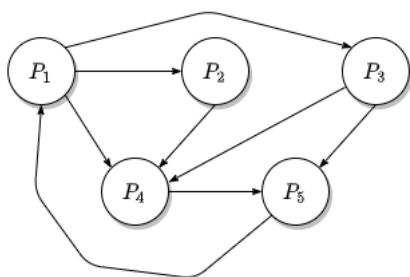


FIGURA 4: Grafo

	Página	PageRank
1º	$P_5$	0.27969961
2º	$P_1$	0.26774467
3º	$P_4$	0.24083375
4º	$P_2$	0.10586099
5º	$P_3$	0.10586099

TABLA 3: Clasificación

En el ejemplo anterior se comprueba la importancia del enlace saliente de  $P_4$  capaz de colocar a una página en el primer puesto del ranking. De hecho si cambiamos este enlace y ponemos que salga de  $P_4$  y llegue a  $P_1$  convertimos a  $P_1$  en la primera del ranking.

### Ejemplo 4

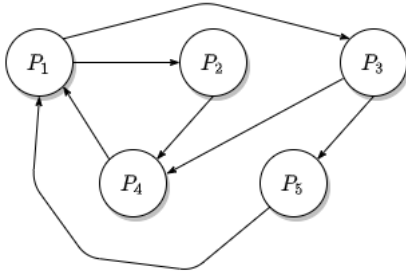


FIGURA 5: Grafo

	Página	PageRank
1º	$P_1$	0.32225461
2º	$P_4$	0.24287173
3º	$P_2$	0.16695821
4º	$P_3$	0.16695821
5º	$P_5$	0.10095724

TABLA 4: Clasificación

Para conseguir que  $P_4$ , la página con más enlaces entrantes, se posicione la primera de la clasificación le quitamos los enlaces salientes.

### Ejemplo 5

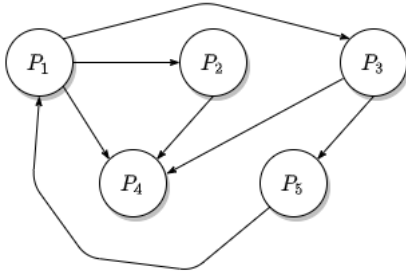


FIGURA 6: Grafo

	Página	PageRank
1º	$P_4$	$2.79048247e - 13$
2º	$P_1$	$2.47850051e - 13$
3º	$P_5$	$1.86281319e - 13$
4º	$P_2$	$1.31449230e - 13$
5º	$P_3$	$1.31449230e - 13$

TABLA 5: Clasificación

Hemos visto que para el algoritmo puntúa más un enlace que sale de una página “importante” que muchos enlaces de páginas “poco importantes”. En el siguiente ejemplo vemos que la página que más enlaces tiene es  $P_3$  y la segunda  $P_4$ . Siguiendo el razonamiento anterior  $P_3$  sería una página “importante” y su enlace valdría mucho, tanto que es capaz de colocar a cualquier página en primera posición. En este caso  $P_3$  cita a  $P_5$  convirtiéndola en una página “importante” y colocándola en primer lugar en la clasificación. Sin embargo, en este ejemplo añadimos un enlace desde  $P_5$ , página importante, a  $P_3$  que también es importante, el resultado es un ajustado desempate ( $1.48851482e - 06$  frente a  $1.37701780e - 06$ ) en el que resulta ganador  $P_5$ .



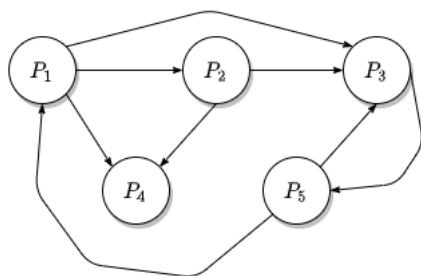
**Ejemplo 6**

FIGURA 7: Grafo

	Página	PageRank
1º	$P_5$	$1.48851482e - 06$
2º	$P_3$	$1.37701780e - 06$
3º	$P_1$	$8.80201594e - 07$
4º	$P_4$	$6.61515329e - 07$
5º	$P_2$	$4.46763904e - 07$

TABLA 6: Clasificación

Por último, nos preguntamos si las dos páginas tienen el mismo número de enlaces entrantes  $P_3$  y  $P_4$ , quedando así “empatados” y una de ellas no tiene enlaces salientes  $P_4$ . El resultado vuelve a ser ajustado, ganando  $P_5$  (gracias al enlace desde  $P_3$ ), en segundo puesto quedaría  $P_3$  y en tercer lugar  $P_4$ . Por lo que no tener enlaces salientes solo te coloca en primer lugar si tienes más enlaces entrantes que las demás páginas.

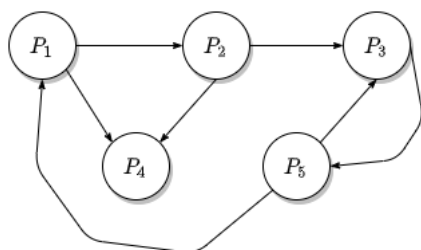
**Ejemplo 7**

FIGURA 8: Grafo

	Página	PageRank
1º	$P_5$	$7.66066766e - 09$
2º	$P_3$	$6.60140578e - 09$
3º	$P_4$	$5.18659092e - 09$
4º	$P_1$	$4.86047008e - 09$
5º	$P_2$	$3.44565523e - 09$

TABLA 7: Clasificación

## 13. Conjunto de datos

Para implementar el algoritmo de PageRank se ha utilizado la conjunto de datos **PMSC-UGR** con 26759991 artículos. Esta conjunto de datos basada en artículos científicos de MEDLINE/PubMed pero también utilizando SCopus herramienta de Elsevier. En PSMC-UGR se ha hecho el esfuerzo de quitar las ambigüedades de los nombres de autores, para ello se ha establecido el ORCID, un número que identifica a cada autor de manera unívoca. Cada archivo esta estructurado de la siguiente forma:

- *PubMedID*. Es un identificador único del artículo proporcionado por PubMed.
- *Journal*. Es el nombre de la revista donde el artículo ha sido publicado.
- *ArticleTitle*. Es el título completo del artículo en inglés.
- *Abstract*. Resumen del artículo.
- *AuthorList*. Contiene información sobre los autores del artículo. Por cada uno podemos encontrar:
  - *LastName*. Contiene el apellido o el nombre único utilizado.
  - *ForeName*. Contiene el resto del nombre.
  - *Identifier*. Es un identificador único asociado con el nombre (ORCID).
- *MeshHeadingList*. Es un vocabulario controlado por NLM, encabezados de temas médicos (Mesh). Se utiliza para caracterizar el contenido del artículo utilizando descriptores de ese tesauro.
- *KeywordList*. Contiene términos controlados en palabras clave que también describen el contenido del artículo.

A estos datos se añaden las citas, el 66.78% de los artículos tienen alguna cita. La media de citas por artículo es de 7 y el total de citas es 3593931. El numero de citas por articulo sigue una típica distribución *ley de potencia* donde pocos artículos tienen muchas citas y muchos artículos tienen pocas citas. Por ejemplo, hay 116365 artículos que tienen solo una cita y hay un artículo que tiene 713 citas.

Además esta conjunto de datos ha sido procesada añadiendo las referencias de cada

### 13. Conjunto de datos

artículo, es decir, cada artículo conoce quien lo ha citado. Este dato no se utiliza en el algoritmo ya que por lo general no se suele conocer, cada artículo solo sabe que artículos ha citado y no llega a conocer los otro artículos que lo han citado.

## 14. Modelos de Recuperación de Información

La Recuperación de Información (RI) busca acceder a contenidos mediante modelos que posteriormente se diseñan e implementan. Su finalidad es estimar la relevancia de los datos a la necesidad de información de un usuario la cual se expresa en una consulta. En nuestro caso, buscamos acceder a la información de nuestro conjunto de datos formada por una selección de documentos. A través de una consulta buscamos mostrar información relevante para el usuario que la ha introducido.

Según el libro [4] podemos definir un modelo de RI como una cuádrupla  $[D, Q, F, R(q_i, f_j)]$ , donde:

- $D$  es el conjunto de representaciones de los documentos.
- $Q$  es el conjunto de representaciones de necesidades de información de los usuarios.
- $F$  es el conjunto de representaciones de las relaciones entre documentos y consultas.
- $R(q_i, d_j)$  es la función de selección y ordenación que permite entregar al usuario unos documentos (ordenados o no) en respuesta a una consulta.

### 14.1. Modelo binario

Este modelo se basa en la teoría de conjuntos y la lógica booleana, como ya comentamos anteriormente solo tiene en cuenta la presencia de alguna de las palabras de la consulta en el documento, no tiene en cuenta la frecuencia ni la importancia de la palabra.

Según el libro [4] definimos el conjunto de todos los términos de la colección de documentos como  $V = \{t_1, t_2, \dots, t_M\}$  y el conjunto de todos los documentos de la colección como  $D = \{d_1, d_2, \dots, d_N\}$ . Cada documento  $d_j$  lo representamos como un subconjunto de  $V$  donde se encuentran los términos que aparecen en ese documento. Además para cada término  $t_i$  se define el conjunto  $T_i$  formado por los documentos que contienen el término  $t_i$  por lo tanto  $T_i$  es un subconjunto de  $D$ .

En este modelo se representan las consultas mediante expresiones booleanas en las cuales se utilizan los operadores booleanos AND, OR y NOT.

El operador AND es un operador binario (necesita dos términos) que indica que se desea buscar los documentos que contentan ambos términos. Por ejemplo, si el usuario escribe “perro AND gato” está buscando los documentos que contienen tanto la palabra “perro” como la palabra “gato”. Formalmente lo que indica es que se realiza la intersección de dos conjuntos, el conjunto formado por los documentos que contienen la palabra “perro” ( $T_1$ ) y los que tienen la palabra “gato” ( $T_2$ ).

Si por ejemplo tuvieramos los siguientes cuatro documentos:

- $d_1 = \{\text{gato, gato, gato, tortuga, pez}\}$
- $d_2 = \{\text{perro, caballo}\}$
- $d_3 = \{\text{gato, perro, águila}\}$
- $d_4 = \{\text{pez, tortuga, tortuga}\}$

Entonces la consulta obtendría el siguiente resultado:

$$\text{perro AND gato} = T_1 \cap T_2 = \{d_2, d_3\} \cap \{d_1, d_3\} = \{d_3\}$$

El operador OR también binario, indica que se desea buscar los documentos que contengan un término o el otro. Siguiendo el ejemplo anterior si el usuario escribe “perro OR gato” está buscando los documentos que contengan la palabra “perro”, o que contengan la palabra “gato” o que contenga ambas. Formalmente lo podemos representar como la unión de los conjuntos formados por los documentos que contienen la palabra “perro” ( $T_1$ ) y los que tienen la palabra “gato” ( $T_2$ ).

$$\text{perro OR gato} = T_1 \cup T_2 = \{d_2, d_3\} \cup \{d_1, d_3\} = \{d_1, d_2, d_3\}$$

Por último, el operador NOT es un operador unario (solo necesita un término) e indica que se desean buscar todos los documentos que no contengan ese término. Este operador se suele combinar con los dos anteriores ya que hace referencia a un conjunto de documentos muy amplio. Este operador se puede representar como la diferencia de conjuntos, el primero conjunto formado por todos los documentos  $D$  y el segundo conjunto formado por las palabras que contienen el término. Por ejemplo:

$$\text{perro AND NOT gato} = T_1 \cap T_2 = \{d_2, d_3\} \cap \{d_2, d_4\} = \{d_2\}$$

$$\text{perro OR NOT gato} = T_1 \cup T_2 = \{d_2, d_3\} \cup \{d_2, d_4\} = \{d_2, d_3, d_4\}$$

Estos operadores se pueden anidar formando consultas más complejas. Por ejemplo, si llamamos, al igual que antes,  $T_1$  a los documentos que tienen la palabra “perro”,  $T_2$  a los que tienen la palabra “gato” y llamamos ahora  $T_3$  a los documentos que contienen la palabra “tortuga”, la siguiente consulta tendría el resultado:

$$\begin{aligned} (\text{tortuga OR gato}) \text{ AND perro} &= (T_3 \cup T_2) \cap T_1 = (\{d_1, d_4\} \cup \{d_1, d_3\}) \cap \{d_2, d_3\} = \\ &= \{d_1, d_3, d_4\} \cap \{d_2, d_3\} = \{d_3\} \end{aligned}$$

El orden de los documento del resultado vendría dado por el PageRank, calculado previamente.

Una de las ventajas de este modelo es su sencillez de implementación, además posee una gran flexibilidad a la hora de realizar las distintas consultas gracias al uso de los operadores lógicos.

Por otro lado, solo se tiene en cuenta la presencia del término de la consulta en cada uno de los documentos. Esto puede ser un error ya que hay términos que cobran más importancia que otros y cuya aparición en el documento debería aumentar la relevancia de éste. Por otro lado, tampoco se tiene en cuenta la frecuencia de cada término en los documentos, no es lo mismo que la palabra aparezca solo una vez que aparezca más de una. Estos inconvenientes se resuelven en el siguiente modelo.

## 14.2. Modelo vectorial

En este modelo se construye un vector por cada documento y consulta en donde se reflejan los términos de cada una de ellas. Es decir, se crea un espacio vectorial de tamaño el número de términos de la colección de documentos.

Según el libro [4], definimos  $V$  y  $D$  al igual que en el modelo anterior y definimos un documento  $d_j$  como un vector de términos  $d_j = (w_{1,j}, w_{2,j}, \dots, w_{M,j})$  donde  $w_{i,j}$  indica el peso del término  $i$  en el documento  $j$ . Este peso valdrá cero si el término no está presente en el documento y en caso de que aparezca tendrá peso positivo.

Con los datos anteriores creamos una matriz de pesos en donde representamos por filas todos los términos y por columnas todos los documentos, quedando la siguiente estructura:

$$\begin{array}{ccccccc} & d_1 & d_2 & \cdots & d_j & \cdots & d_N \\ \begin{array}{l} t_1 \\ t_2 \\ \vdots \\ t_M \end{array} & \left[ \begin{array}{cccccc} w_{1,1} & w_{1,2} & \cdots & w_{1,j} & \cdots & w_{1,N} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,j} & \cdots & w_{2,N} \\ \vdots & \vdots & & \vdots & & \vdots \\ w_{M,1} & w_{M,2} & \cdots & w_{M,j} & \cdots & w_{M,N} \end{array} \right] \end{array}$$

Este modelo se basa en dos conceptos fundamentales:

- Esquema de pesos.
- Similitud entre dos vectores de términos (por ejemplo, entre un documento y una consulta).

La similitud entre dos vectores de términos  $q$  y  $d$  la calcularemos mediante la distancia coseno que se define según el libro [4] como:

$$sim(q, d) = \cos(\alpha) = \frac{\sum_{i=1}^M w_{i,q} \cdot w_{i,d}}{\sqrt{\sum_{i=1}^M w_{i,d}^2} \cdot \sqrt{\sum_{i=1}^M w_{i,q}^2}} = \frac{q}{|q|} \cdot \frac{d}{|d|}$$

Por otro lado, a la hora de calcular el peso de un término en cada documento debemos tener en cuenta dos conceptos:

- La frecuencia del término en el documento ya que cuanto más se repita más importancia tendrá este término en el documento.
- La especificidad del término en el documento; es decir, cuanto menos usual sea la palabra dentro de la colección de documentos más peso tendrá esta al considerarse relevante su aparición.

Utilizaremos el modelo *tf.idf*, mencionado en el libro [4], formado por dos componentes que representan los conceptos explicados anteriormente:

- *Term frequency* (tf): indica la frecuencia del término en el documento.
- *Inverse document frequency* (idf): número de documentos en donde aparece referenciado el término pero a la inversa. Se da mayor peso a los términos que se encuentran en un número menor de documentos.

Por lo tanto, calcularemos el peso de un término de la siguiente forma:

$$w_{i,j} = tf_{i,j} \cdot idf_i = tf_{i,j} \cdot \log \left( \frac{N}{n_i} \right)$$

Donde  $tf_{i,j}$  es la frecuencia del término  $i$  en el documento  $j$ ,  $N$  es el número de documento que tiene la colección y  $n_i$  es el número de documentos en donde aparece el término  $i$ .

Tras calcular estos pesos se normaliza cada vector  $d_j$ , es decir, dividimos el vector por su longitud:

$$d_j = \frac{d_j}{|d_j|} \text{ donde } |v| = \sqrt{\sum_{i=1}^M v_i^2}$$

Por último, el procedimiento consistirá en calcular los pesos de la consulta  $q$  y calcular la similitud de la consulta con los demás documentos. Tras estas operaciones se obtiene un vector de relevancia donde queda reflejada la similitud de cada documento a la consulta, estas similitudes son multiplicadas por el vector del PageRank para obtener el nuevo orden.

Retomamos el anterior ejemplo, donde el conjunto  $D$  estaba formado por los documentos:

- $d_1 = \{\text{gato, gato, gato, tortuga, pez}\}$
- $d_2 = \{\text{perro, caballo}\}$
- $d_3 = \{\text{gato, perro, águila}\}$
- $d_4 = \{\text{pez, tortuga, tortuga}\}$

Por tanto, los términos son  $V = \{\text{gato, tortuga, pez, perro, caballo, águila}\}$ , y supongamos que la consulta es  $q = \{\text{gato, tortuga}\}$ . La matriz  $tf$  sería la siguiente:

$tf$	$d_1$	$d_2$	$d_3$	$d_4$	$q$
gato	3	0	1	0	1
tortuga	1	0	1	2	1
pez	1	0	0	1	0
perro	0	1	1	0	0
caballo	0	1	0	0	0
águila	0	0	1	0	0



Mostramos a continuación la matriz  $w$  ya normalizada:

$tf$	$d_1$	$d_2$	$d_3$	$d_4$	$q$
gato	0.94	0	0.40	0	0.92
tortuga	0.13	0	0.17	0.64	0.38
pez	0.31	0	0	0.77	0
perro	0	0.45	0.40	0	0
caballo	0	0.89	0	0	0
águila	0	0	0.81	0	0

El resultado de calcular la similitud de la consulta entre todos los documentos se muestra en la siguiente tabla:

sim	$d_1$	$d_2$	$d_3$	$d_4$
$q$	0.92	0	0.43	0.24

Por último, se multiplicaría este vector con el vector del PageRank componente a componente.

### 14.3. Técnicas de modificación de la consulta

Dentro del modelo vectorial buscamos personalizar las consultas, es decir, que dependiendo del usuario que realice la consulta se muestre antes los documentos más relevantes para ese usuario. Para ello empleamos la pseudo-realimentación de consultas la cual busca expandir la consulta utilizando los primeros documentos más relevantes para ello.

Una de las mayores ventajas de esta técnica es que se trata de un proceso automático, no necesita de la intervención del usuario. Sin embargo, la efectividad de la técnica depende de la bondad del primer resultado, si la consulta devuelve un resultado donde los documentos son relevantes la extensión de la consulta conseguirá recuperar los documentos más relevantes, pero si por el contrario la primera recuperación no es buena aparecerán en la consulta expandida términos relacionados con ella por lo que esta será menos relevante aún para el usuario.

El objetivo de esta técnica consiste en conseguir una consulta expandida  $q'$  a partir de

una consulta  $q$ , para ello se utiliza la fórmula de Rocchio extraída del libro [4]:

$$q' = \alpha q + \frac{\beta}{D_r} \sum_{d_j \in D_r} d_j - \frac{\gamma}{D_{nr}} \sum_{d_j \in D_{nr}} d_j$$

Donde  $D_r$  es el conjunto de documentos que previamente se han fijado como relevantes para el usuario y  $D_{nr}$  son los documentos no relevantes. Mientras que  $\alpha$ ,  $\beta$  y  $\gamma$  son parámetros que hay que fijar. Usualmente se le da valores  $\alpha = 1$ ,  $\beta = 0.75$  y  $\gamma = 0$ , ya que no buscamos ampliar la consulta con documentos no relevantes.

Tras calcular la nueva consulta  $q'$  se vuelve a calcular la similitud entre esta y los documentos, obteniendo un vector que refleja la similitud de la nueva consulta  $q'$  y los documentos, este vector se multiplica por el vector del PageRank para obtener el nuevo orden.

Reutilizamos el ejemplo anterior donde la matriz de pesos era:

$tf$	$d_1$	$d_2$	$d_3$	$d_4$	$q$
gato	0.94	0	0.40	0	0.92
tortuga	0.13	0	0.17	0.64	0.38
pez	0.31	0	0	0.77	0
perro	0	0.45	0.40	0	0
caballo	0	0.89	0	0	0
águila	0	0	0.81	0	0

Aplicamos la ecuación de Rocchio con parámetros  $\alpha = 1$ ,  $\beta = 0.75$  y  $\gamma = 0$  suponiendo que los documentos que tienen interés para el usuario son, por ejemplo  $d_1$  y  $d_3$ :

	$1 \cdot q$	$(0.75/2) \cdot d_1$	$(0.75/2) \cdot d_3$	Rocchio
gato	0.92	0.35	0.15	1.42
tortuga	0.38	0.05	0.06	0.49
pez	0	0.12	0	0.12
perro	0	0	0.15	0.15
caballo	0	0	0	0
águila	0	0	0.30	0.30

A continuación nos quedaríamos con los 10 ó 15 términos con mayor valor, pero como

#### 14. Modelos de Recuperación de Información

en este caso solo tenemos 6 términos, nos quedaremos con los 4 mayores, por lo tanto  $q'$  tendría el siguiente valor:

	$q'$
gato	1.42
tortuga	0.49
pez	0
perro	0.15
caballo	0
águila	0.30

La similitud de la nueva consulta  $q'$  es:

sim	$d_1$	$d_2$	$d_3$	$d_4$
$q'$	0.91	0.04	0.62	0.20

Vemos que al contrario que en la anterior consulta el documento  $d_2$  obtiene algo de relevancia (0.04) esto es debido a que uno de los documentos importantes para el usuario es el  $d_3$  y ambos contienen el término “perro”. Por último, al igual que en el método anterior este vector de relevancias se multiplicará término a término por el vector de PageRank.

## 15. Interfaz gráfica

Para mostrar mejor los resultados del buscador y mejorar la comunicación entre el usuario y el sistema se ha desarrollado una interfaz gráfica cuya apariencia es la siguiente:

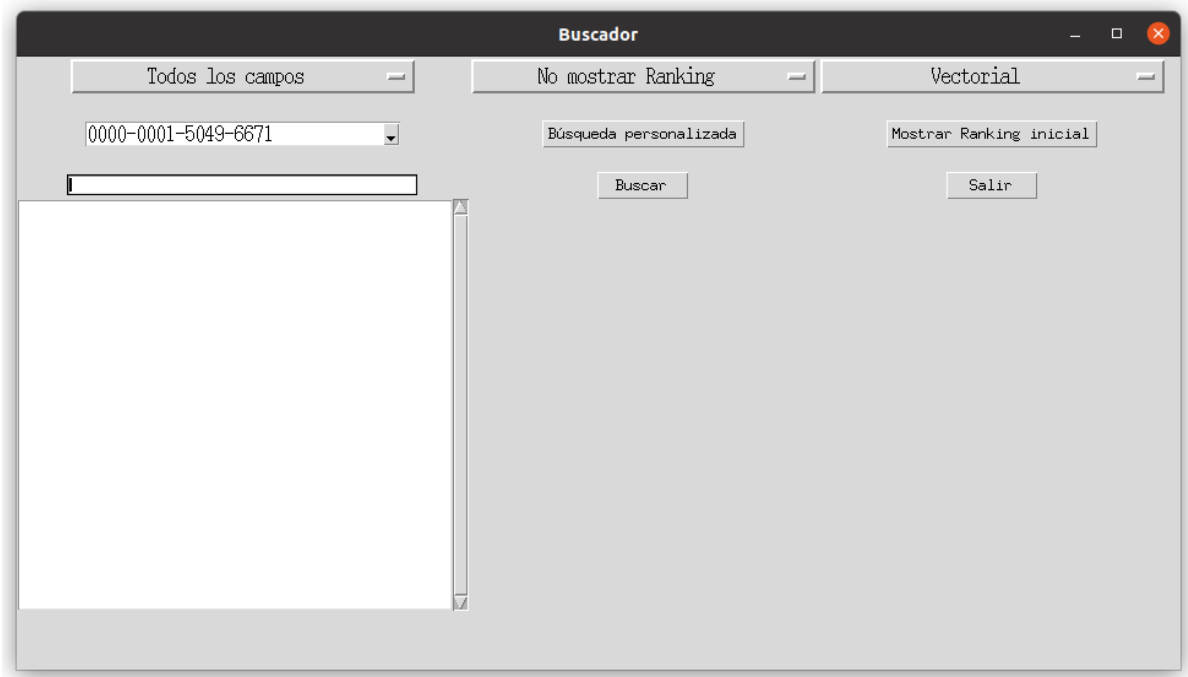


FIGURA 9: Interfaz gráfica del buscador.

En el desplegable situado en la parte superior a la izquierda se puede seleccionar el modelo que deseamos utilizar (Vectorial o Booleano).



FIGURA 10: Desplegable para seleccionar el modelo.

Como ya explicamos anteriormente dependiendo del modelo las consultas se realizan de

forma distinta:

- **Modelo booleano.** En este modelo se puede elegir en que parte del documento se quiere realizar la búsqueda: título, abstract, palabras clave o todas las partes. Este dato se indica en el desplegable situado en la parte superior izquierda.

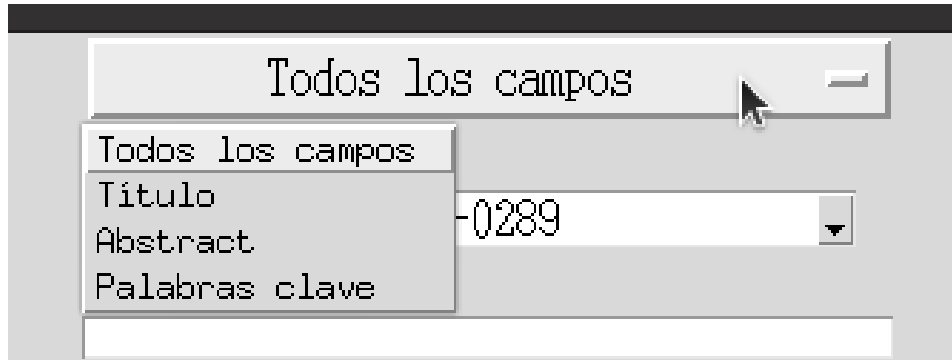


FIGURA 11: Desplegable para seleccionar la parte del documento en donde se desea realizar la búsqueda.

Una vez seleccionado en que parte del documento deseamos realizar la búsqueda, introducimos la consulta con las correspondientes expresiones lógicas y pulsamos en el botón “Buscar”. Por ejemplo, introducimos la consulta “cognitive AND papers”.

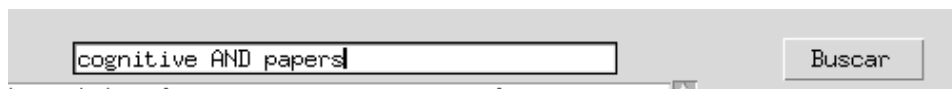


FIGURA 12: Celda donde se introduce la consulta y botón que dispara la búsqueda.

El resultado de la búsqueda se mostrará en el espacio situado justo debajo de la celda donde se introduce la consulta.

## 15. Interfaz gráfica

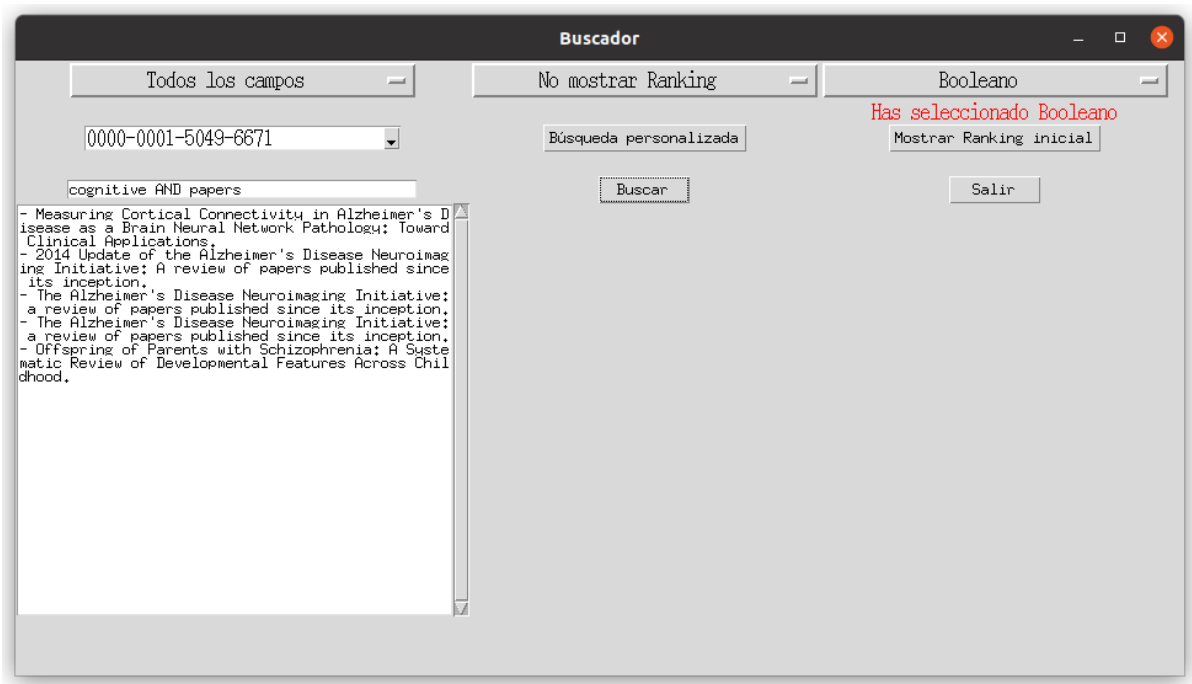


FIGURA 13: Resultado de la búsqueda en el modelo booleano.

- **Modelo vectorial.** En este modelo podemos elegir entre realizar una búsqueda normal o una búsqueda personalizada. Para realizar una búsqueda normal solo hay que introducir la consulta en la celda y pulsar el botón “buscar” (en este modelo no se utilizan esquemas lógicos). Por ejemplo, podemos buscar las palabras “extension network”.

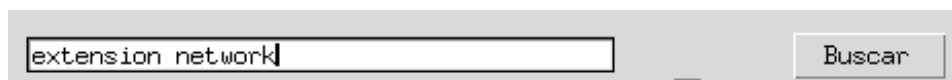


FIGURA 14: Celda donde se introduce la consulta y botón que dispara la búsqueda.

Cuyo resultado sería:

## 15. Interfaz gráfica

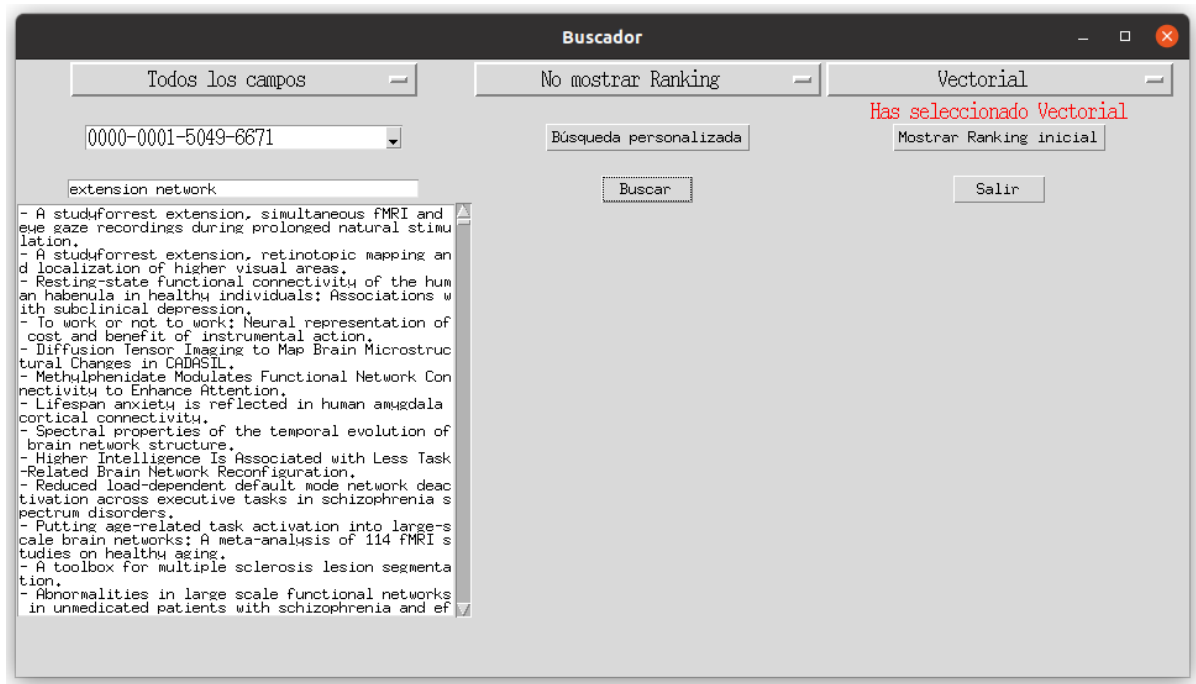


FIGURA 15: Resultado de la búsqueda en el modelo vectorial (búsqueda normal).

Para realizar una búsqueda personalizada, primero debemos elegir el usuario que la realiza. Como ya se ha comentado anteriormente los usuarios son los propios autores de los documentos, que han sido previamente identificados por un ORCID. Para elegir que usuario realiza la consulta se utiliza el desplegable situado justo encima de la celda de la consulta y se selecciona uno de los ORCID.

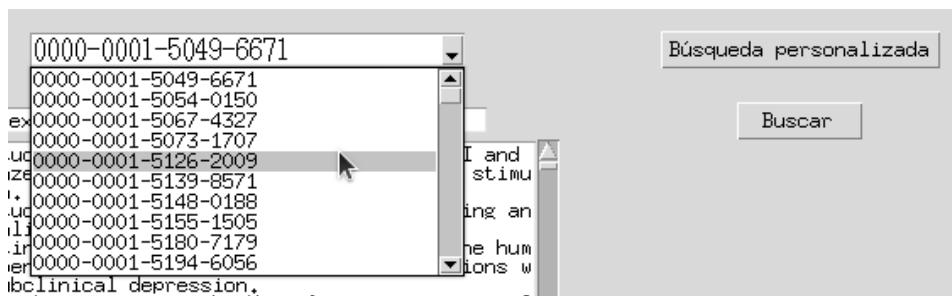
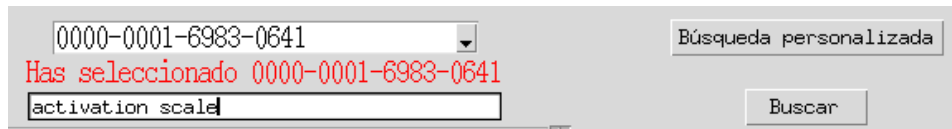


FIGURA 16: Desplegable para elegir el ORCID del autor que realiza la búsqueda.

Por ejemplo, seleccionamos el autor 0000 – 0001 – 6983 – 0641, introducimos la consulta de nuevo en la celda y pulsamos el botón “Búsqueda personalizada” para disparar la búsqueda. Por ejemplo, realizamos la consulta “activation scale”.

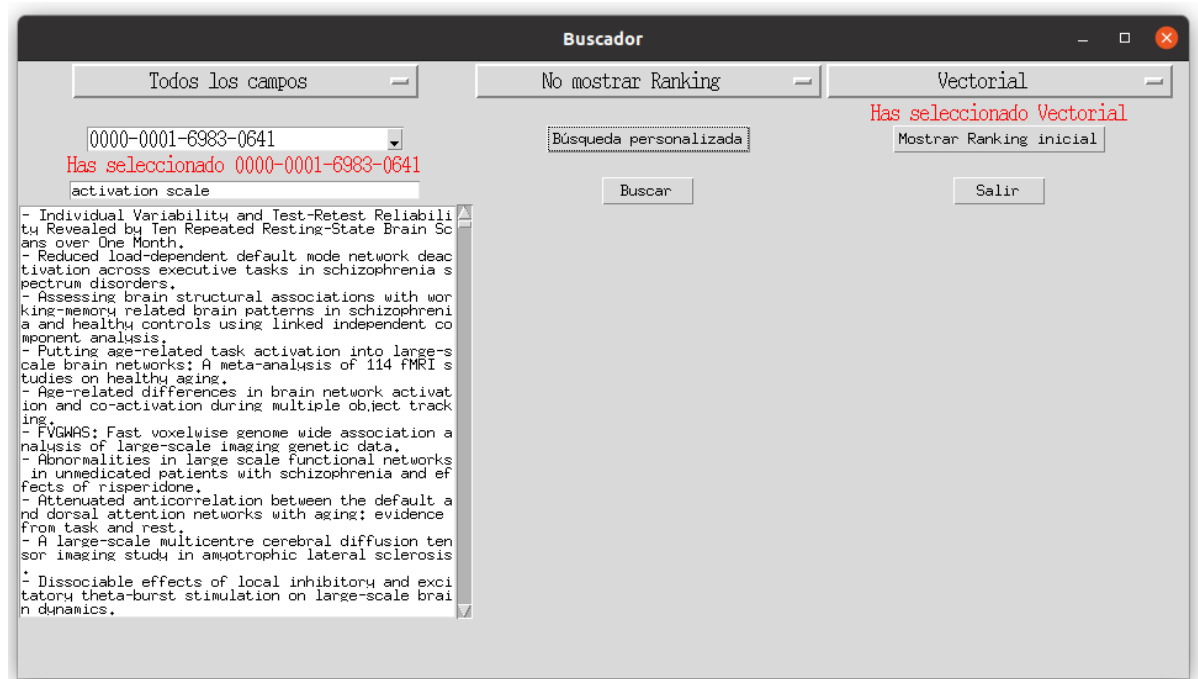
## 15. Interfaz gráfica



0000-0001-6983-0641  
Has seleccionado 0000-0001-6983-0641  
activation scale  
Búsqueda personalizada  
Buscar

FIGURA 17: Celda donde se introduce la consulta y botón que dispara la búsqueda.

Cuyo resultado es el siguiente:



Buscador

Todos los campos No mostrar Ranking Vectorial

0000-0001-6983-0641  
Has seleccionado 0000-0001-6983-0641  
activation scale

Búsqueda personalizada Mostrar Ranking inicial

Buscar Salir

- Individual Variability and Test-Retest Reliability Revealed by Ten Repeated Resting-State Brain Scans over One Month.  
- Reduced load-dependent default mode network deactivation across executive tasks in schizophrenia spectrum disorders.  
- Assessing brain structural associations with working-memory related brain patterns in schizophrenia and healthy controls using linked independent component analysis.  
- Putting age-related task activation into large-scale brain networks: A meta-analysis of 114 fMRI studies on healthy aging.  
- Age-related differences in brain network activation and co-activation during multiple object tracking.  
- FVGWAS: Fast voxelwise genome wide association analysis of large-scale imaging genetic data.  
- Abnormalities in large scale functional networks in unmedicated patients with schizophrenia and effects of risperidone.  
- Attenuated anticorrelation between the default and dorsal attention networks with aging: evidence from task and rest.  
- A large-scale multicentre cerebral diffusion tensor imaging study in amyotrophic lateral sclerosis  
- Dissociable effects of local inhibitory and excitatory theta-burst stimulation on large-scale brain dynamics.

FIGURA 18: Resultado de la búsqueda en el modelo vectorial (búsqueda personalizada).

Además podemos mostrar el resultado del algoritmo PageRank, pulsando el botón “Mostrar Ranking Inicial”. Este botón es independiente del modelo en el que se encuentre el buscador.



## 15. Interfaz gráfica

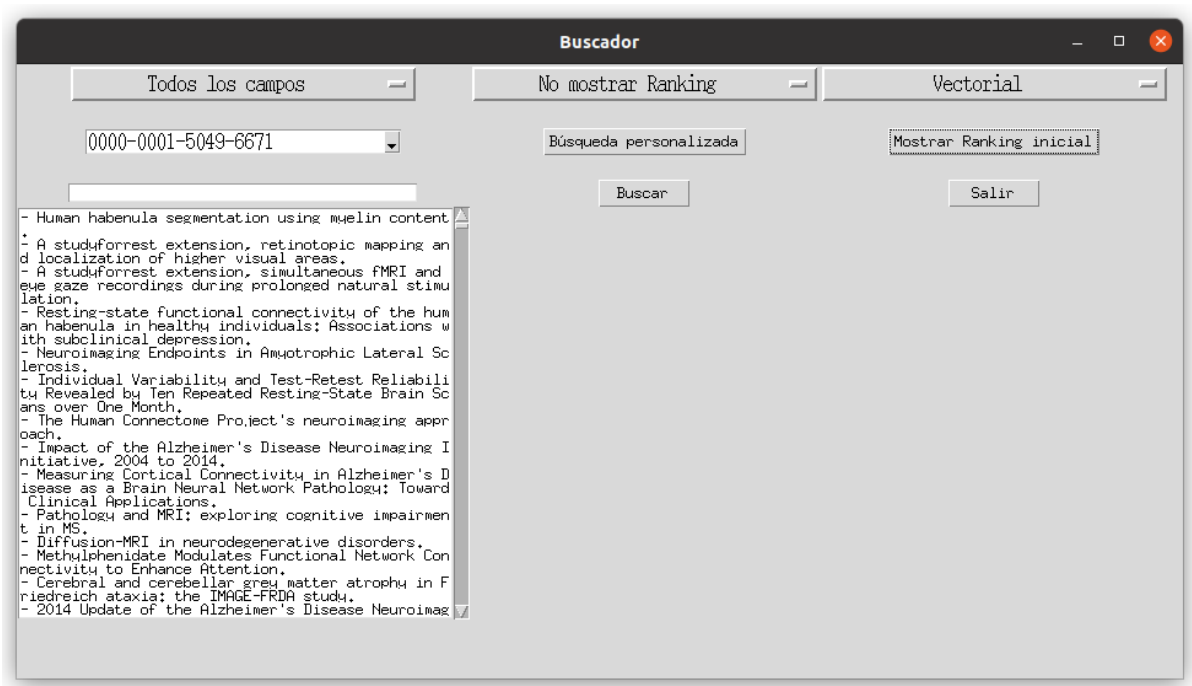


FIGURA 19: Resultado del algoritmo PageRank.

Por último, en cualquier consulta podemos mostrar los pesos de esta, para ello se selecciona en la pestaña “Mostrar Ranking” para mostrarlo y “No mostrar Ranking” para ocultarlo.



FIGURA 20: Desplegable para elegir si mostrar el ranking o no.

Los pesos se mostraría de la siguiente forma.

## 15. Interfaz gráfica

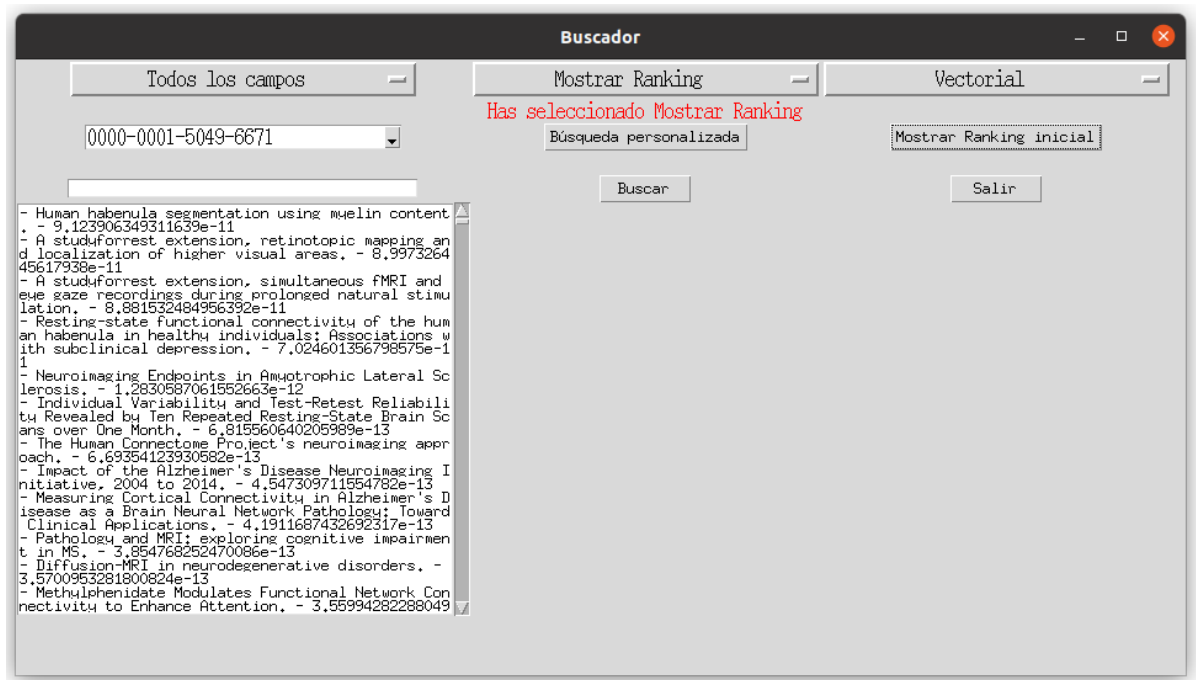


FIGURA 21: Ranking inicial con los pesos del ranking.

## 16. Esquema de las consultas

Como se ha comentado anteriormente según el modelo en el se encuentre el sistema la consulta debe introducirse de una forma o de otra.

- **Modelo booleano.** En este modelo se debe seguir el esquema de la lógica booleana la cual cumple las siguientes reglas:
  - Los operadores binarios (AND y OR) deben aparecer entre dos términos. En caso de que la consulta no lo cumpla se mostrará el siguiente error.



FIGURA 22: Mensaje de error en el operador lógico.



FIGURA 23: Mensaje de error en el operador lógico.

- En el caso del operador unario NOT el término a consultar debe ir precedido por el operador. En caso contrario se mostrará el siguiente error.



FIGURA 24: Mensaje de error en el operador lógico.

- Al combinar los dos operadores, el operador unario debe ir precedido del operador binario, en caso contrario también se mostrará un mensaje de error.



FIGURA 25: Mensaje de error en el operador lógico.

- En el caso de anidar esquemas lógicos y de utilizar paréntesis, deberán aparecer el paréntesis de abrir y el de cerrar. En caso contrario, se mostrará un mensaje de error.

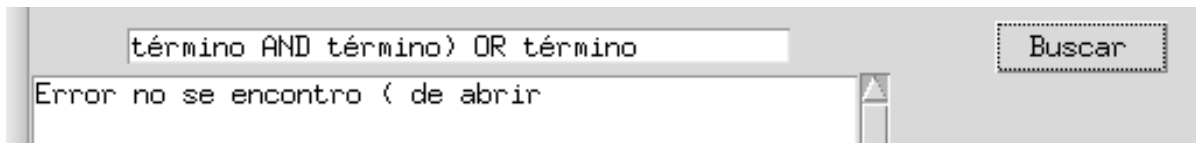


FIGURA 26: Mensaje de error en los paréntesis.

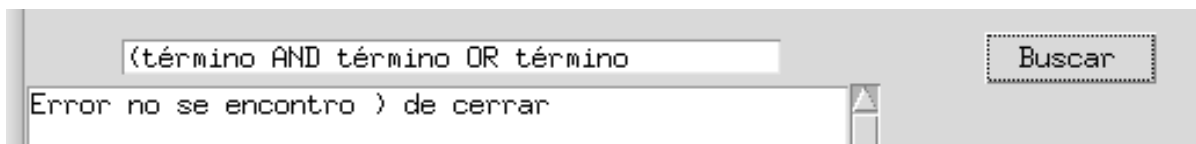


FIGURA 27: Mensaje de error en los paréntesis.

- En el caso de querer introducir un término compuesto en la consulta este deberá introducirse entre comillas dobles. En el caso de que no se introduzcan las comillas en alguno de los dos extremos del término se mostrará el error por pantalla.

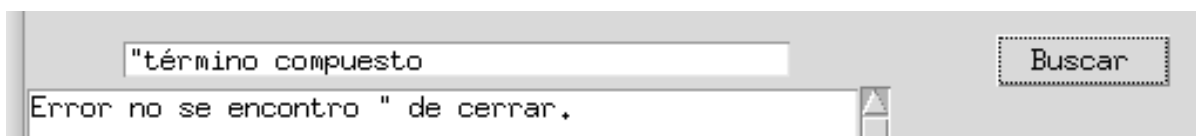


FIGURA 28: Mensaje de error en las comillas dobles.

- **Modelo vectorial.** En este modelo las consultas no tienen ninguna restricción. También se pueden introducir términos compuestos, como en el anterior modelo, introduciendo estos términos entre comillas, en el caso de que el usuario se olvide de cerrar las comillas el sistema mostrará un mensaje de error (Figura 28).

## 17. Resultados

En esta sección mostraremos los resultados de varias ejecuciones para verificar que estos son lógicos. Realizaremos distintas pruebas dependiendo de las condiciones de cada modelo.

Para ilustrar mejor los ejemplos elegimos dos archivos, por ejemplo el quinto y el décimo clasificados según el PageRank. Estos son:

### Quinto clasificado

- Título: Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis.
- ORCID autor: 0000-0003-0267-3180
- Abstract: Amyotrophic lateral sclerosis (ALS) is a progressive neurodegenerative, clinically heterogeneous syndrome pathologically overlapping with frontotemporal dementia. To date, therapeutic trials in animal models have not been able to predict treatment response in humans, and the revised ALS Functional Rating Scale, which is based on coarse disability measures, remains the gold-standard measure of disease progression. Advances in neuroimaging have enabled mapping of functional, structural, and molecular aspects of ALS pathology, and these objective measures may be uniquely sensitive to the detection of propagation of pathology in vivo. Abnormalities are detectable before clinical symptoms develop, offering the potential for neuroprotective intervention in familial cases. Although promising neuroimaging biomarker candidates for diagnosis, prognosis, and disease progression have emerged, these have been from the study of necessarily select patient cohorts identified in specialized referral centers. Further multicenter research is now needed to establish their validity as therapeutic outcome measures.
- Keywords: 'Amyotrophic lateral sclerosis' 'biomarker.' 'magnetic resonance imaging' 'motor neuron disease' 'trial'

### Decimo clasificado

- Título: Pathology and MRI: exploring cognitive impairment in MS.
- ORCID autor: 0000-0002-6378-0070

- **Abstract:** Cognitive impairment is a frequent symptom in people with multiple sclerosis, affecting up to 70 % of patients. This article reviews the published association of cognitive dysfunction with neuroimaging findings. Cognitive impairment has been related to focal T2 hyperintense lesions, diffuse white matter damage and cortical and deep gray matter atrophy. Focal lesions cannot sufficiently explain cognitive dysfunction in MS; microstructural tissue damage detectable by diffusion tensor imaging and gray matter atrophy are probably at least as relevant. Resting state functional magnetic resonance imaging is increasingly used to investigate the contribution of functional connectivity changes to cognitive function in MS. The fact that at least one third of MS patients are not overtly cognitively impaired despite significant radiographic tissue damage argues for protective factors (brain reserve, cognitive reserve) that require further clarification. It is concluded that the reported correlations between imaging findings and cognitive function do not imply causality. Well conceived and sufficiently powered longitudinal studies are lacking. Such studies would help unravel protective mechanisms against cognitive decline and identify suitable imaging techniques to monitor cognitive function in individual patients with MS.
- **Keywords:** 'cognitive impairment' 'functional connectivity' 'gray matter' 'magnetic resonance imaging' 'multiple sclerosis' 'white matter'
- **Modelo booleano.** Elegimos, por ejemplo, una palabra que esté en los dos documentos, por ejemplo, *sclerosis* y otra palabra que solo aparezca en uno de ellos, por ejemplo, *patients* que solo se encuentra en el segundo documento. Realizamos las siguientes búsquedas.

En primer lugar buscamos solo la palabra *sclerosis* y nos aseguramos de que se muestran los dos documentos. Vemos que efectivamente el documento “Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis” aparece en primera posición, mientras que el documento “Pathology and MRI: exploring cognitive impairment in MS” aparece en tercera posición.

## 17. Resultados

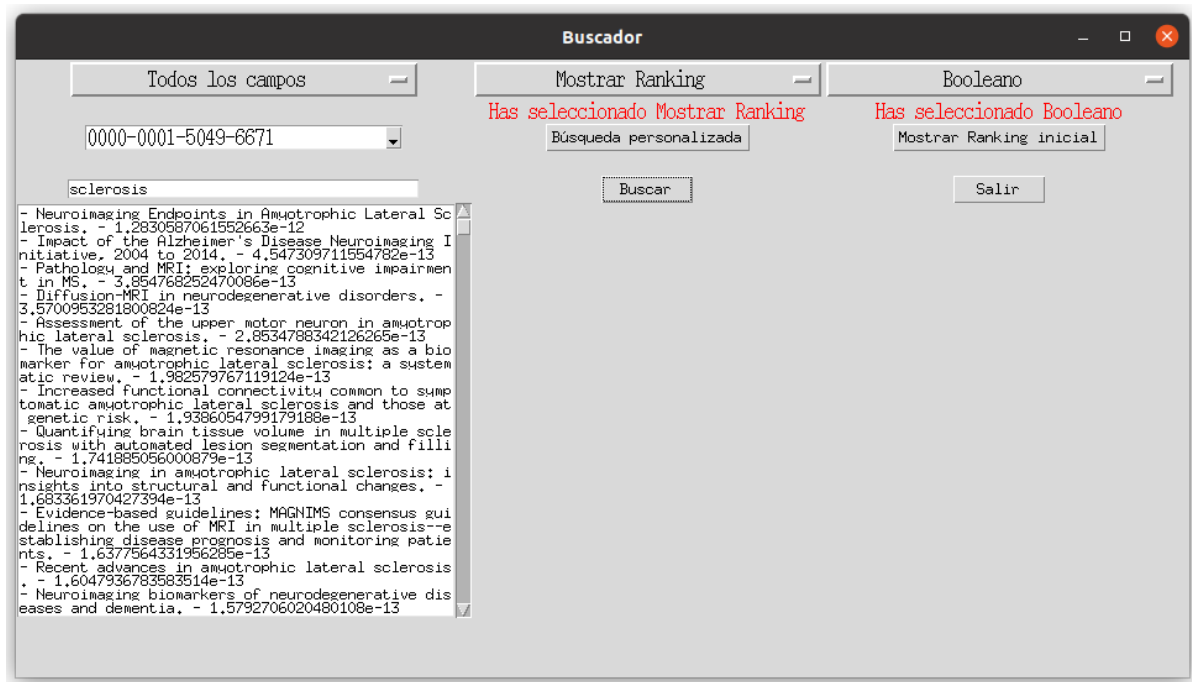


FIGURA 29: Resultado de la consulta *sclerosis*.

A continuación realizamos la consulta *sclerosis AND patients*. Y comprobamos que efectivamente de los dos documentos solo aparece el documento “Pathology and MRI: exploring cognitive impairment in MS”.

## 17. Resultados

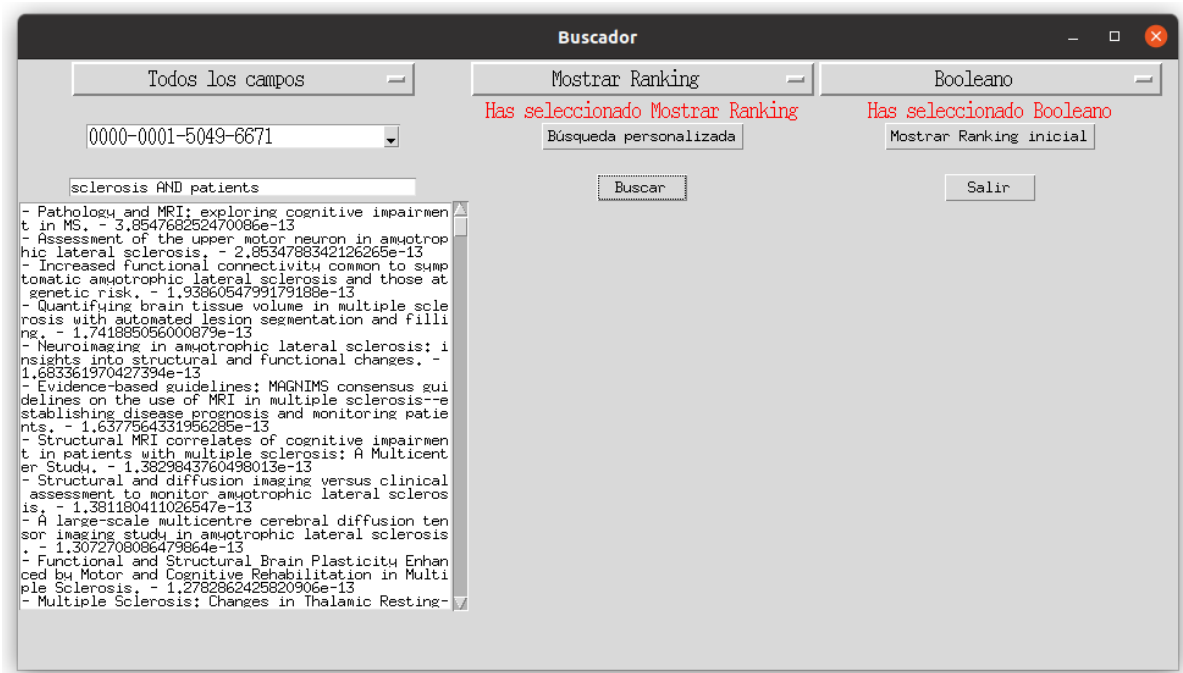


FIGURA 30: Resultado de la consulta *sclerosis AND patients*.

Probamos a realizar la consulta *sclerosis AND NOT patients*. Comprobamos ahora que de los dos documentos solamente aparece “Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis”



## 17. Resultados

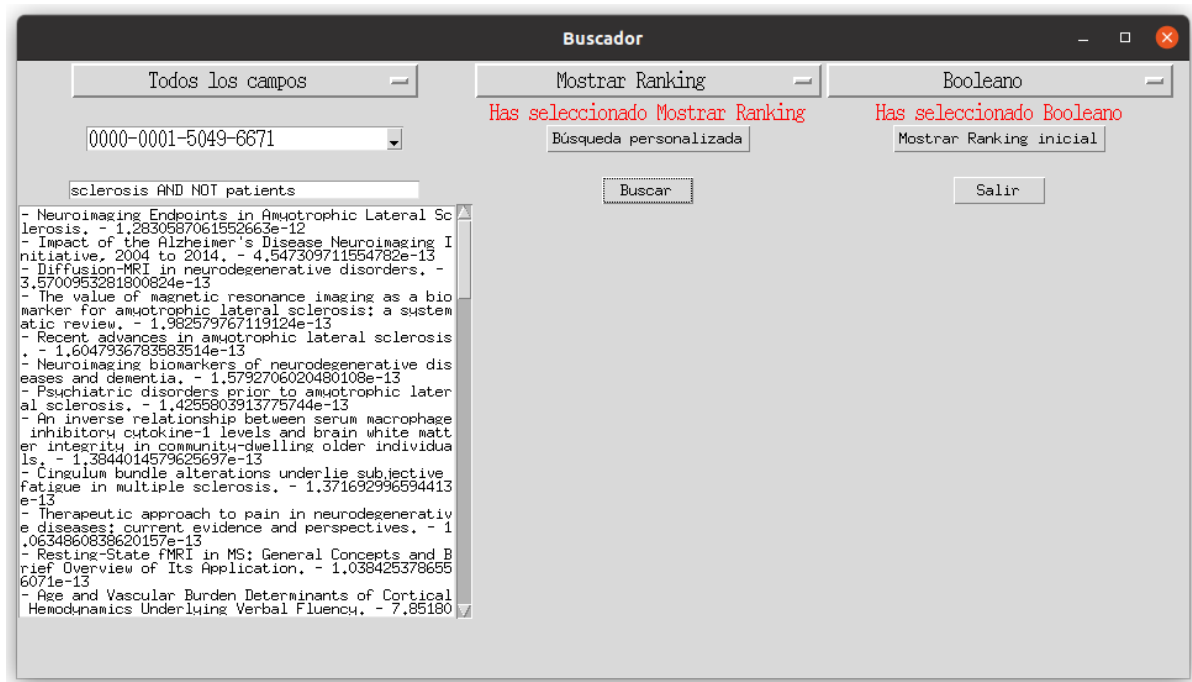


FIGURA 31: Resultado de la consulta *sclerosis AND NOT patients*.

Elegimos ahora una palabra que solo se encuentre en el documento “Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis”, por ejemplo, *neurodegenerative*, y realizamos la consulta *neurodegenerative OR patients*. Comprobamos que efectivamente aparecen los dos documentos, “Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis” situado en la primera posición y “Pathology and MRI: exploring cognitive impairment in MS” en la tercera.

## 17. Resultados

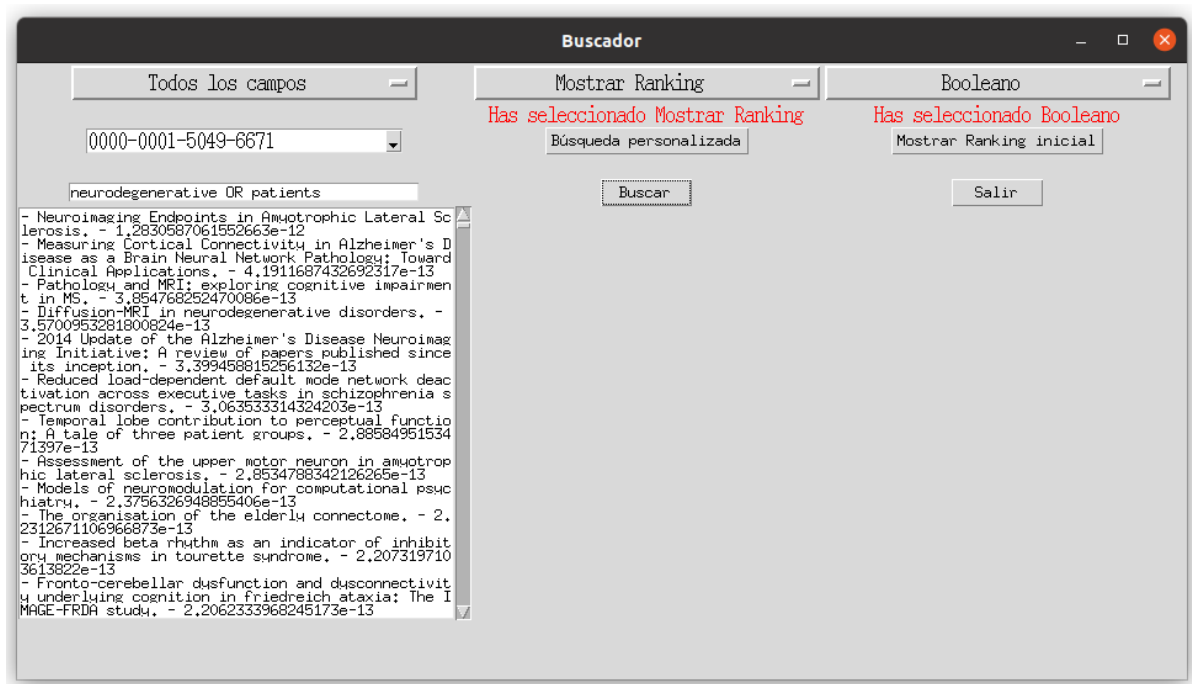


FIGURA 32: Resultado de la consulta *neurodegenerative OR patients*.

A continuación seleccionamos que el sistema solo haga la búsqueda en el Título, realizamos la consulta *Sclerosis OR Pathology* y efectivamente aparecen los dos documentos. Uno situado el primero del ranking y el otro situado en tercer lugar.

## 17. Resultados

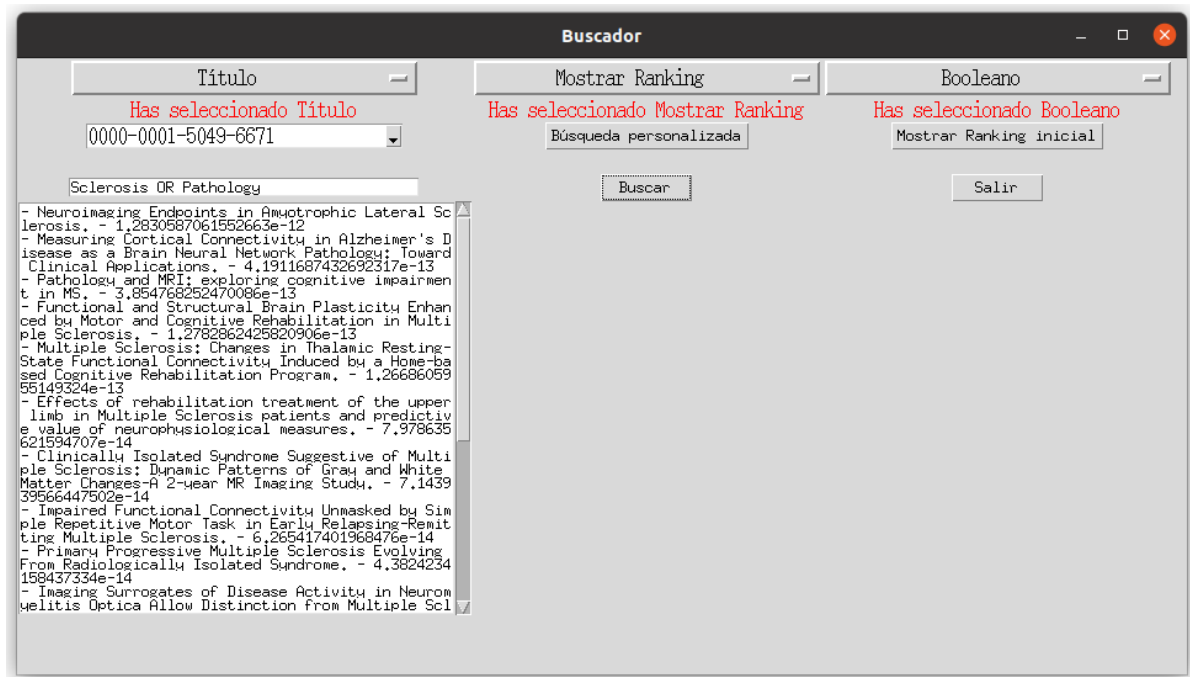


FIGURA 33: Resultado de la consulta *Sclerosis OR Pathology*.

- **Modelo vectorial.** Para realizar las pruebas en este modelo utilizaremos los mismos documentos. Realizamos la búsqueda *sclerosis patients* y comprobamos que efectivamente aparecen los dos documentos encabezando el ranking. Vemos que aunque el documento “Pathology and MRI: exploring cognitive impairment in MS” contiene las dos palabras, su peso no es suficiente como para superar al otro documento (que solo contiene una palabra), puesto que el peso del orden del PageRank tiene más fuerza (recordemos que es el quinto posicionado según el PageRank mientras que “Pathology and MRI: exploring cognitive impairment in MS” es el décimo).

## 17. Resultados

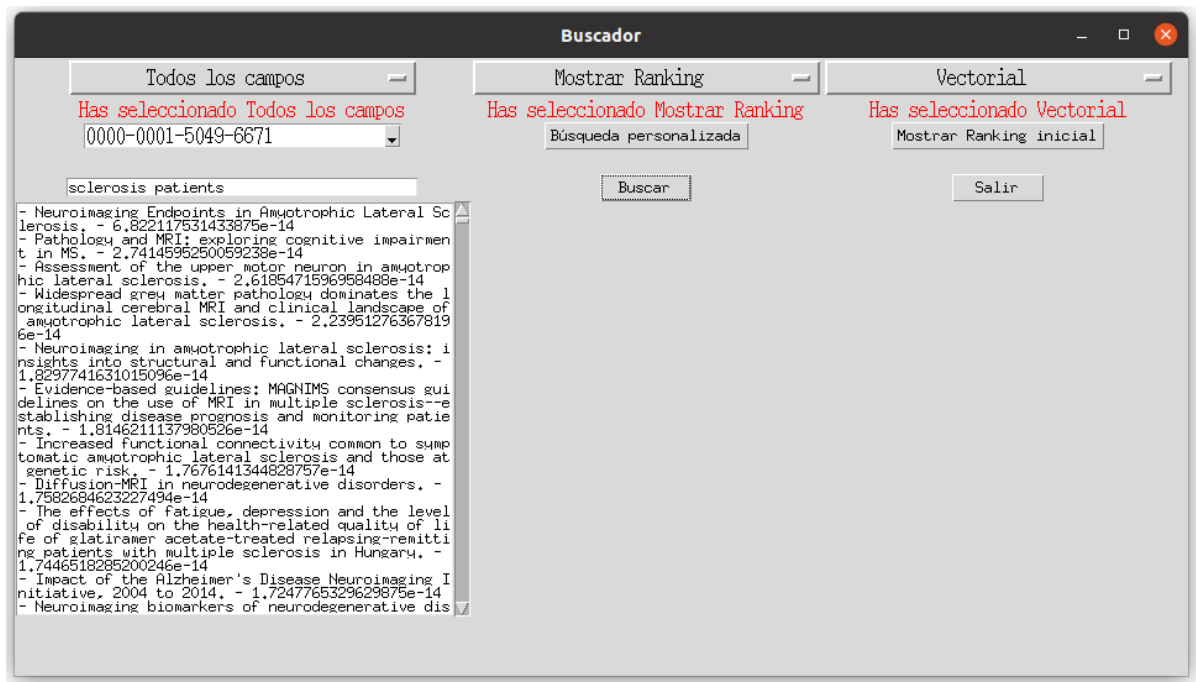


FIGURA 34: Resultado de la consulta *sclerosis patients*.

Si a la consulta anterior le añadimos el término Pathology conseguimos situar el documento “Pathology and MRI: exploring cognitive impairment in MS” por encima de “Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis” ya que el primero contiene todas las palabras mientras que el otro solo contiene la palabra *sclerosis*.

## 17. Resultados

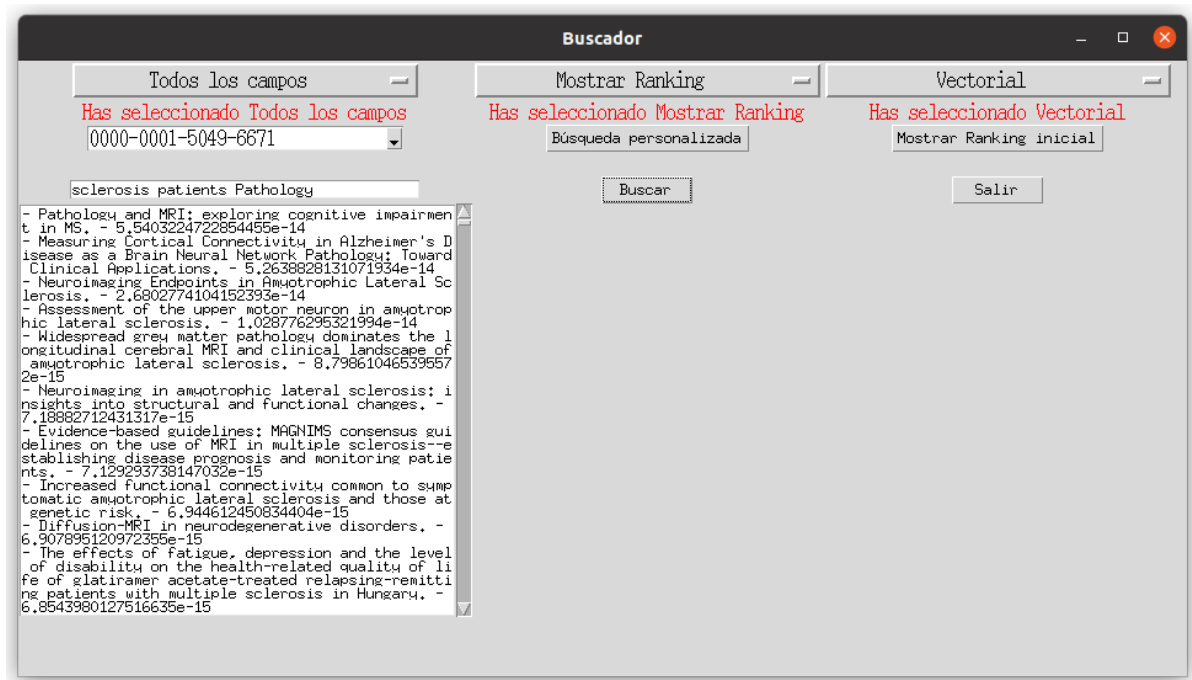


FIGURA 35: Resultado de la consulta *sclerosis patients Pathology*.

Por último, si realizamos la consulta anterior siendo el usuario 0000 – 0003 – 0267 – 3180, es decir, el autor de “Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis.” vemos como este se sitúa por delante del otro documento aún conteniendo menos palabras de la consulta.

## 17. Resultados

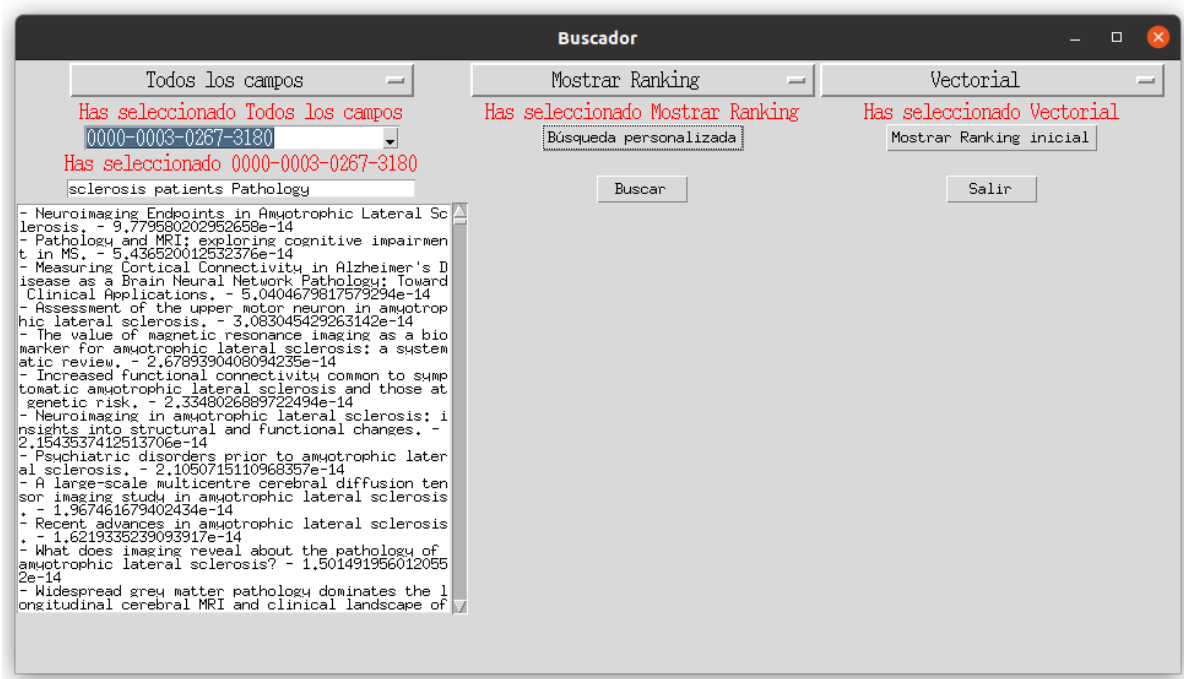


FIGURA 36: Resultado de la consulta *sclerosis patients Pathology*.

A continuación mostramos los pesos de las distintas consultas del modelo vectorial.

Consulta	Título	PageRank	Similitud	Resultado final
"sclerosis patients"	Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis	1.283e-12	0.0532	6.822e-14
	Pathology and MRI: exploring cognitive impairment in MS	3.855e-13	0.0711	2.741e-14
"sclerosis patients Pathology"	Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis	1.283e-12	0.021	2.680e-14
	Pathology and MRI: exploring cognitive impairment in MS	3.855e-13	0.144	5.540e-14
"sclerosis patients Pathology" (personalizada)	Neuroimaging Endpoints in Amyotrophic Lateral Sclerosis	1.283e-12	0.076	9.780e-14
	Pathology and MRI: exploring cognitive impairment in MS	3.855e-13	0.141	5.437e-14

## 17. Resultados

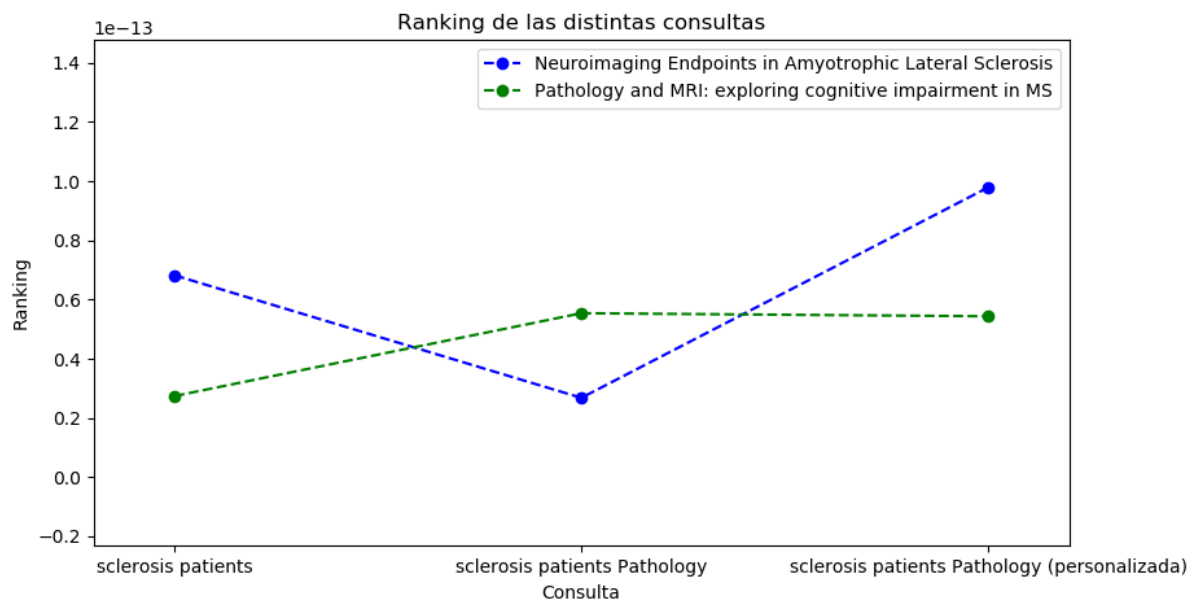


FIGURA 37: Ranking de las consultas del modelo vectorial.

## **18. Conclusiones y trabajos futuros**

Concluimos que el algoritmo PageRank obtiene resultados bastante buenos a la hora de ordenar los archivos por relevancia. Además los modelos implementados consiguen buenos resultados, consiguiendo personalizar las búsquedas.

Sería interesante, como futuros proyectos, crear perfiles de usuario con distintos intereses para poder personalizar mejor la búsqueda consiguiendo mostrar en los primeros resultados los archivos más interesantes para el usuario dependiendo de su perfil.



## **19. Sistema basado en la información**

Un sistema de acceso a la información es un sistema dotado de un conjunto de técnicas para buscar, categorizar, modificar y acceder a la información que se encuentra en un sistema: conjunto de datos, bibliotecas, archivos o internet.

¿¿ Poner esto en algún lado ??

## Referencias

- [1] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Siam. 2000.
- [2] Luis Merino y Evangelina Santos. *Álgebra lineal con métodos elementales*. Paraninfo. 1999.
- [3] Rafael Ortega Rios. *Modelos matemáticos*. Editorial Universidad de Granada. 2013.
- [4] Fidel Cacheda Seijo, Juan Manuel Fernández Luna y Juan Francisco Huete Guadix. *Recuperación de Información. Un enfoque práctica y multidisciplinar*. RA-MA Editorial y Publicaciones. 2011.