

Importancia de Aeropuertos por Vuelos

Claudio Agustín Armas



Universidad
Nacional
de Córdoba



Facultad
de Matemática,
Astronomía, Física
y Computación

20 de noviembre de 2021

1. Introducción

En junio de 2014, la base de datos de rutas de **OpenFlights** registró 67663 rutas entre 3321 aeropuertos en 548 aerolíneas en todo el mundo [1]. En base a los datos registrados, se plantean las siguientes preguntas; ¿Cuáles son los aeropuertos que tienen mayor importancia por vuelos?, ¿Cuál es el aeropuerto con menor importancia?, ¿Existe algún método que permita clasificar a un conjunto de elementos en función de su importancia?. Cabe aclarar que la importancia de un aeropuerto no significa que tuvo la mayor cantidad de vuelos recibidos sobre los demás, sino que influyen tanto los vuelos de llegada como los vuelos de salida del mismo. Un problema de este tipo se lo puede ver como un problema de clasificación. Entonces, una idea para resolver este problema es construir un modelo matemático que permita encontrar de manera iterativa la mejor clasificación. De este modo, el objetivo principal del presente informe es “determinar” (aproximar) los aeropuertos con las mejores clasificaciones.

Se han propuesto muchos esquemas de clasificación para diferentes problemas de este tipo. Este esquema es un modelo para obtener una relación de orden en un conjunto finito dado. Por ejemplo, PageRank es un algoritmo empleado por Google para clasificar los sitios web en los resultados de búsqueda. Fue desarrollado por Larry Page y Sergey Brin los fundadores de Google, en 1996 como parte de un proyecto de investigación.

El modelo de PageRank es lineal, es por eso que un paso importante en el algoritmo de iteración de Page y Brin se basa en la aplicación del teorema de Perron-Frobenius. El teorema es un resultado clásico del álgebra lineal, que tiene mucha importancia en los problemas de clasificación, en particular en el método de PageRank [2]. La idea del algoritmo es medir la importancia de las páginas del sitio web en función de cuántas y qué tan “importantes” están vinculadas otras páginas a ellas. Según Google: “*PageRank funciona contando el número y la calidad de los enlaces a una página para determinar una estimación aproximada de la importancia del sitio web.*”.

Es fácil ver por qué el escenario específico donde se aplica el modelo de clasificación no es tan importante, es decir, porque las situaciones puede ser diferentes pero en cualquier caso se tiene un conjunto de elementos interactuando entre sí con la propiedad razonable de que el rango de un elemento es alto si interactúa con elementos de alto rango. Por supuesto, esto no es suficiente por sí solo para obtener un rango alto. Parece que es evidente que la definición de las interacciones y la forma en que se traduce estas interacciones en una formula matemática es un paso crucial en el esquema de clasificación y esto afectará profundamente el resultado de la clasificación.

2. Planteo del problema

El problema de clasificación puede formularse fácilmente como un problema de autovalor lineal. Esto se puede explicar mediante el siguiente ejemplo.

Supongamos que hay un concurso, con un número N de participantes a los que se quiere asignar una puntuación. La puntuación se basa en las interacciones con otros participantes. Entonces es conveniente definir un vector de valores de clasificación r , con componentes positivas r_j , que indica la fortaleza del j -ésimo participante. La definición de la puntuación es crucial ya que, influye completamente en el comportamiento del modelo. Se puede definir una puntuación para el i -ésimo participante como

$$s_i = \frac{1}{n_i} \sum_{j=1}^N a_{ij} r_j$$

donde N es el número total de participantes en el concurso, a_{ij} es un valor no negativo relacionado con el resultado del juego entre el participante i y el participante j , n_i es el número de juegos jugados por el participante i .

La i -ésima puntuación depende de todas las demás puntuaciones mediante un coeficiente que caracteriza la interacción entre i y j . En la definición se toma una clase de normalización; la división por n_i no es importante en el caso de que todos los participantes jueguen el mismo número de juegos, por ejemplo en un campeonato de fútbol tiene importancia en caso de que sea posible juegos adicionales [3]. La definición lleva a un modelo lineal clásico

$$s_i = \sum_{j=1}^N \frac{a_{ij}}{n_i} r_j = \sum_{j=1}^N b_{ij} r_j. \quad (1)$$

A modo de comparación, en el problema de clasificación de sitios web, la puntuación que PageRank otorga a los sitios se basa en el concepto de “popularidad de enlaces”; una determinada página web es importante si, además de recibir enlaces de otras páginas de alto rango, tiene un número limitado de enlaces a otras páginas. Una representación formal del concepto viene dada por la fórmula

$$r(P) = \sum_{Q \rightarrow P} \frac{r(Q)}{|Q|}, \quad (2)$$

donde P y Q son páginas de la web, $r(P)$ y $r(Q)$ los rangos de las páginas, es decir, su importancia en la web, $|Q|$ es el número de enlaces externos de la página Q y “ $Q \rightarrow P$ ” debajo de la sumatoria significa que la suma se extiende sobre las páginas Q que tienen un enlace a la página P .

Para obtener una clasificación de P , las páginas enlazadas a P deben tener un rango alto, además hay que tener en cuenta que la relevancia de estas páginas se reduce por el número total de enlaces que tienen. Entonces, cuantos menos enlaces externos tenga una página, mejor será para la clasificación de P .

La formula (2) puede escribirse de manera similar a (1). Sea P_1, P_2, \dots, P_n las n páginas de la web. Podemos definir la matriz de preferencia de la siguiente manera

$$r(P_i) = \sum_{P_j \rightarrow P_i} \frac{r(P_j)}{|P_j|} = \sum_{P_j \rightarrow P_i} \frac{1}{|P_j|} \cdot r(P_j) = \sum_{j=1}^N a_{ij} r(P_j) \quad (3)$$

donde

$$a_{ij} = \text{Prob}(P_j \rightarrow P_i) = \begin{cases} \frac{1}{|P_j|} & \text{si } P_j \rightarrow P_i \\ 0 & \text{en otro caso.} \end{cases} \quad (4)$$

Como se indica aquí, la matriz A puede verse como una matriz de transición, cuyos elementos son probabilidades de transición de una página a otra. Esta configuración asume que si P_j está vinculado a P_i , la transición de P_j a P_i es solo una de las posibles transiciones, cada una con la misma probabilidad. Entonces la clasificación de sitios web tiene la misma estructura matemática que el ejemplo del concurso, solo difieren en como se definen los coeficientes.

Ahora se debe plantear la razón de por que se involucran los autovalores. Volviendo al el ejemplo del concurso, es razonable suponer que la puntuación de un participante es proporcional a su fortaleza (el rango), según lo definido en (1). Al llamar a λ la constante de proporcionalidad e indicamos por simplicidad con a_{ij} a los $\frac{a_{ij}}{n_i}$, obtenemos

$$s_i = \lambda r_i$$

reemplazando s_i por lo definido en (1)

$$\sum_{j=1}^N a_{ij} r_j = \lambda r_i,$$

escribiendo la ecuación en forma matricial, obtenemos

$$Ar = \lambda r,$$

es la ecuación que se conoce por definir el autovalor de una matriz y su correspondiente autovector. El significado de esto es: si se asume la relación lineal del puntaje y la proporcionalidad entre los rangos y los puntajes, entonces es posible definir un vector de rango que debe ser un autovector de la matriz de coeficientes A (matriz de preferencia), y el autovalor correspondiente es el escalar para la proporción.

Pasando ahora al problema de clasificación del sitio web y la fórmula correspondiente a (2) que establece como se definen las puntuaciones en el algoritmo de PageRank, ahora aparecerá la razón por la que un autovalor en particular y su correspondiente autovector son importantes.

Supongamos que recopilamos los rangos de todas las páginas web en un vector r y queremos calcularlo de forma iterativa, partiendo de una evaluación previa de r . Si $r^{(k)}$ es la k -ésima iteración del vector r , al principio la configuración más razonable (no la única posible) es asumir que todas las páginas tienen el mismo rango y, teniendo en cuenta una clase de normalización, puede comenzar desde

$$r^{(0)} = \left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n} \right)^T.$$

En el contexto de clasificación de sitios web, el concepto de probabilidad ya surgió de forma bastante natural en la definición de la matriz A . Está claro que también se puede interpretar el vector de rango de forma probabilística. El vector inicial $r = r^{(0)}$ puede verse como “el vector de distribución de probabilidad uniforme”; al comienzo del proceso de clasificación, todas las páginas web tienen la misma probabilidad.

La ecuación general en (2) la escribimos en términos de las n páginas P_1, P_2, \dots, P_n de la web, en particular para la i -ésima página P_i . Entonces para la primera iteración obtenemos:

$$r^{(1)}(P_i) = \sum_{P_j \rightarrow P_i} \frac{r^{(0)}(P_j)}{|P_j|} = \sum_{P_j \rightarrow P_i} \frac{1}{|P_j|} \cdot r^{(0)}(P_j) = \sum_{j=1}^n a_{ij} \cdot r^{(0)}(P_j) \quad i = 1, 2, \dots, n.$$

En forma matricial la ecuación puede ser escrita como

$$r^{(1)} = Ar^{(0)}.$$

Claramente se define una secuencia iterativa, concretamente

$$r^{(k+1)} = Ar^{(k)} \quad k = 0, 1, 2 \dots \quad (5)$$

Esta es la iteración general del *método de la potencia*. De este modo, el problema de clasificación a llevado a un procedimiento iterativo que involucra el vector de rangos y la matriz de preferencia. El problema matemático esta relacionado con el autovalor más importantes de una matriz, llamado *autovalor dominante*.

3. El autovalor dominante y el método de las potencias

Existen resultados fundamentales que se relacionan con respecto al autovalor dominante de una matriz. El más famoso de estos resultados es probablemente el teorema Perron-Frobenius. La idea de esta sección es presentar el autovalor dominante y su relación con el método de las potencias.

3.1. Autovalor dominante

El autovalor dominante es importante por muchas razones. Un aspecto fundamental teórico, es que el autovalor dominante da un límite para todos los autovalores de la matriz. En el contexto general de la definición de autovalores, donde son números complejos, el autovalor dominante da el radio de círculo donde se encuentran todos los autovalores de la matriz.

Sea una matriz $A \in \mathbb{C}^{n \times n}$ diagonalizable con $V = [v^1 \dots v^n] \in \mathbb{C}^{n \times n}$ tal que

$$V^{-1}AV = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix},$$

con $\lambda_i \in \mathbb{C}$ y $|\lambda_1| \geq \dots \geq |\lambda_n|$ [4]. De este modo, definimos formalmente el autovalor dominante de la siguiente manera.

Definición 1. Se dirá que λ_1 es un *autovalor dominante* de A si $|\lambda_1| > |\lambda_2|$. En este caso, v^1 será llamado el *autovector dominante* de A .

Un asunto importante en la práctica es que no se necesita encontrar todos los autovalores para seleccionar el dominante. Hay métodos que llevan directamente al autovalor dominante. Además, como los autovectores también son importantes, estos métodos llevan al mismo tiempo a encontrar también los autovectores correspondiente al autovalor dominante. Uno de los métodos más empleados para encontrar un autovalor dominante y su correspondiente autovector es el llamado método de las potencias.

3.2. Método de las potencias

Con el fin de encontrar el autovalor dominante de una matriz con su correspondiente autovector se han diseñado una gran variedad de métodos, uno de los más destacados es el método de las potencias.

Sea una matriz $A \in \mathbb{R}^{n \times n}$ que tiene n autovectores linealmente independientes asociados a los autovalores $\lambda_1, \lambda_2, \dots, \lambda_n$ y supongamos que

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$$

Esto dice que λ_1 es el autovalor dominante con multiplicidad uno. Sean v^1, v^2, \dots, v^n los correspondientes autovectores. Esto es

$$Av^i = \lambda_i v^i, \quad i = 1, 2, \dots, n.$$

El método de potencia comienza a partir de cualquier vector $x^{(0)}$ arbitrario, genera una sucesión de vectores $\{x^{(k)}\}$ que converge al autovector asociado al autovalor dominante. Para que la sucesión sea convergente hay que suponer lo siguiente: sea $x^{(0)}$ un vector arbitrario en \mathbb{R}^n . Podemos escribir $x^{(0)}$ como una combinación lineal de v^1, v^2, \dots, v^n que, dada a la hipótesis de independencia lineal, son una base de \mathbb{R}^n .

$$x^{(0)} = \sum_{i=1}^n c_i v^i \quad \text{suponiendo } c_1 \neq 0.$$

comenzando de $x^{(0)}$ se puede generar la siguiente sucesión

$$x^{(1)} = Ax^{(0)}, \quad x^{(2)} = Ax^{(1)}, \quad \dots, \quad x^{(k)} = Ax^{(k-1)}, \quad \dots$$

entonces, el siguiente resultado es válido:

Teorema 1. *Para la sucesión $\{x^{(k)}\}_{k \in \mathbb{N}}$, se tiene que*

$$\lim_{k \rightarrow \infty} \frac{x_j^{(k+1)}}{x_j^{(k)}} = \lambda_1 \quad \text{y} \quad \lim_{k \rightarrow \infty} \frac{x^{(k)}}{x_j^{(k)}} = cv^1. \quad (6)$$

Donde j es un índice por el cual $x_j^{(k)} \neq 0$ para cada valor de k .

Algo a tener en cuenta es que en las implementaciones prácticas del método es probable que surjan algunos problemas numéricos. El método, si se implementa como se ha presentado, da problemas de overflow/underflow. Por esta razón en cada paso es conveniente normalizar el vector $x^{(k)}$. Las propiedades de la convergencia no se modifican y se evita que las normas se vuelvan demasiado grandes.

Cabe mencionar que el límite de la sucesión es un autovector asociado al autovalor dominante. Por lo tanto, para el problema de clasificación de las páginas web, los rangos de las páginas son las componentes de este vector límite, convenientemente normalizados.

4. La importancia del teorema de Perron-Frobenius en el método PageRank

En un problema de clasificación, el teorema de Perron-Frobenius es importante porque da condiciones suficientes para que el problema tenga solución. Las condiciones están claramente en la matriz A . Es como tener propiedades que garanticen que se puede encontrar, un ordenamiento adecuado y/o una clasificación, en un conjunto definido. Las matrices que cumplen con estas condiciones son las matrices irreducibles. A continuación se realiza una breve descripción de las matrices irreducibles que cumplen un papel importante dentro del teorema.

4.1. Matrices irreducibles

Existen muchas formas equivalentes de caracterizar matrices irreducibles, algunas son algebraicas y otras geométricas. Dos de las definiciones más relacionadas con el problema de clasificación son:

Definición 2. Una matriz A es irreducible si no existe una permutación que transforme A en una matriz de bloque de la forma

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

donde A_{11} y A_{22} son matrices cuadradas.

Definición 3. Una matriz A no negativa es irreducible si para cualquier vector $r \geq 0$ se tiene que $Ar \geq 0$

Otra alternativa es definir las matrices irreducibles por medio de matrices primitivas. Una posible definición de una matriz primitiva es la siguiente:

Definición 4. Sea $A \in \mathbb{R}^{n \times n}$ tal que $A \geq 0$. Diremos que A es una matriz primitiva si existe un $m \geq 1$ tal que $A^m > 0$.

Entonces una forma equivalente de definir una matriz irreducible se presenta en la siguiente proposición.

Proposición 1. Sea $A \in \mathbb{R}^{n \times n}$, con $A \geq 0$. Entonces son equivalentes:

- 1 A es irreducible.
- 2 $(I + A)^{n-1} > 0$.
- 3 $I + A$ es primitiva.

En particular se tiene que, si una matriz A es primitiva, entonces es irreducible [5]. Sin embargo, las definiciones están lejos de ser fáciles de usar, principalmente si se trata de matrices muy grandes, y como se mencionó antes, la idea es aplicar el algoritmo de PageRank para un conjunto grande de datos sobre las rutas entre aeropuertos en el mundo, entonces la matriz que se definirá más adelante será demasiado grande y aplicar estas definiciones puede salir muy caro computacionalmente.

En el teorema de Perron-Frobenius una hipótesis fundamental es que la matriz sea irreducible, esto es para poder encontrar un autovector con entradas no negativas y un autovalor positivo.

4.2. Teorema de Perron-Frobenius

Tradicional e históricamente, el teorema puede expresarse en dos enunciados, el primero para matrices positivas y el otro para matrices no negativas. En realidad, la primera forma se debe a Perron. Frobenius, por su parte trataba de extender el teorema en el caso más general de matrices no negativas. Obtuvo la segunda forma, pero con la necesidad de establecer más condiciones, que la matriz sea irreducible. En el caso del teorema de Perron no se garantiza que una matriz positiva (real) tenga solo autovalores reales, ya que pueden ser números complejos en general. Por otra parte el teorema de Frobenius tiene en cuenta una afirmación muy importante, el radio espectral de una matriz, que se define como el máximo de los valores absolutos de sus autovalores. Entonces, para una mayor generalización se enunciará el teorema para matrices no negativas. Un posible enunciado de la extensión del teorema de Perron-Frobenius es el siguiente.

Teorema 2. Perron-Frobenius

Sea $A \in \mathbb{R}^{n \times n}$ una matriz irreducible no negativa con periodo p y radio espectral ρ . Entonces las siguientes afirmaciones son válidas.

- (i) El número ρ autovalor positivo de A , llamado el autovalor de Perron-Frobenius.

- (ii) ρ es simple. Ambos espacios propios, derechos e izquierdo, asociados con ρ son unidimensionales.
- (iii) A tiene un autovector r a derecha y otro autovector s a izquierda. Las componentes de ambos son positivas y son los únicos autovectores con todas las componentes positivas asociados con ρ .
- (iv) La matriz A tiene exactamente p (período) autovalores complejos con módulo ρ . Cada uno de ellos es una raíz simple del polinomio característico y es el producto de ρ con una p -ésima raíz de la unidad.
- (v) $\lim_{k \rightarrow +\infty} \left(\frac{A}{\rho}\right)^k = rs^T$, donde los autovalores están normalizados, entonces $s^T r = 1$. Además, la matriz rs^T es la proyección sobre el espacio propio V_ρ , llamado proyección de Perron.
- (vi) El autovalor Perron-Frobenius ρ satisface la desigualdad

$$\min_i \sum_j a_{ij} \leq \rho \leq \max_i \sum_j a_{ij}.$$

La tesis no se puede obtener sin la hipótesis de que A es irreducible. La diferencia del teorema (2) con respecto al de Perron es la existencia de un único autovalor dominante real, pero hay valores propios complejos con el mismo valor absoluto.

Ahora la idea es establecer por qué se debe aplicar el teorema de Perron-Frobenius al problema de clasificación. Para la solución del problema de clasificación, el autovalor es solo una constante de proporcionalidad que relaciona la puntuación y el rango. La parte importante de la solución es que exista un vector de clasificación y , por supuesto, las clasificaciones deben ser números positivos. Entonces la importancia radica en tener una solución positiva, donde el teorema (2) establece un autovalor positivo y un autovector positivo correspondiente.

Finalmente, la no unicidad de la solución en el caso no negativo es bastante teórico, porque se debe a soluciones complejas. En una necesidad práctica de tener un esquema de clasificación, las soluciones complejas no son de interés.

5. Caso de aplicación

Establecidas las herramientas teóricas para poder aplicar el método de PageRank, en esta sección se presenta el caso de aplicación para el conjunto de datos registrados por OpenFlights, donde se empleará todo lo presentado hasta ahora, usando como herramienta de lenguaje a python. A medida que se vaya presentando el código de python, se irá explicando y describiendo cada paso que se realizó para llegar a la solución.

En primer lugar, se importa el paquete numpy para las emplear las funciones que se necesiten para el código python e importamos a los datos sobre los aeropuertos y las rutas, que contiene OpenFlights.

```
import numpy as np

datos_aeropuertos = np.genfromtxt("../Proyecto/airports.dat.txt", delimiter=",",
dtype="str", encoding="Latin-1")
```

`datos_aeropuertos` contiene información de 7698 aeropuertos más populares del mundo, donde nos interesa principalmente las columnas [1, 2, 3, 4] que contienen:

- El nombre del aeropuerto.
- La ciudad.
- El país.
- Código IATA de 3 letras. Nulo si no está asignado, y se lo representa como \N.

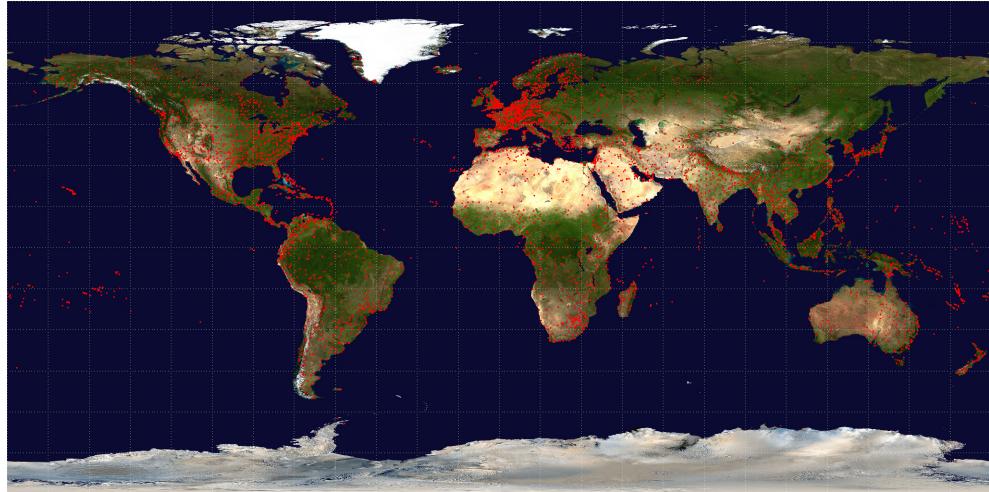


Figura 1: Mapa global de los aeropuertos (puntos rojos) en el mundo [1].

```
# 'datos_rutas' contiene 67663 rutas entre aeropuertos del mundo.

datos_rutas = np.genfromtxt("../Proyecto/routes.dat.txt", delimiter=",",
                           dtype="str", encoding="Latin-1")

rutas = datos_rutas[:, [2, 4]]
```

El archivo `rutas` es una matriz de 67663 filas por 2 columnas, que contiene códigos de las columnas 2 y 4 de `datos_rutas`, donde la primera columna representa los códigos de los aeropuertos de salida y la segunda son códigos de aeropuertos de llegada.



Figura 2: Mapa global de las rutas entre los aeropuertos en el mundo [1].

```

aeropuertos = np.unique(rutas)
#En la lista 'aeropuertos' se utiliza np.unique para obtener de forma unica los
# codigos que intervienen en 'rutas'.

rutas_indice = rutas.copy()
for idx in range(len(aeropuertos)):
    rutas_indice = np.where(rutas_indice == aeropuertos[idx], idx, rutas_indice)

#Se construye una matriz de indices, de la misma dimension de 'rutas',
# que devuelve los indices de los codigos que tienen en 'aeropuertos'.

rutas_indice = rutas_indice.astype(int)

#Se toma la parte entera de los indices.

```

Ahora se debe construir la matriz de enlaces, donde se vinculan los distintos aeropuertos.

```

m = len(aeropuertos)
matriz_enlaces = np.zeros((m, m))

#Se construye una matriz de ceros de orden m x m, la cantidad de elementos
# en 'aeropuertos'.

m_0, n_0 = rutas_indice.shape

```

Se toma la dimensión de `rutas_indice`, donde nos interesa `m_0` ya que, por medio de un `for` se tomará cada fila `i` de `rutas_indice` para obtener los índices `[rutas_indice[i, 1], rutas_indice[i, 0]]` que permiten ubicar las coordenadas de los códigos en la matriz de enlaces, y cada posición que se encuentre pisarlo con 1.

```

for i in range(m_0):

    matriz_enlaces[rutas_indice[i, 1], rutas_indice[i, 0]] = 1

#Los indices rutas_indice[i, 0] y rutas_indice[i, 1], son las posiciones
# del aeropuerto de salida y el aeropuerto de llegada, respectivamente.

```

En el siguiente `for`, se suma los unos que tiene cada columna de la matriz de enlace. Cabe aclarar que no todas las columnas tienen unos, existen algunas solo con ceros.

```

for j in range(m):
    w = np.sum(matriz_enlaces[:, j])
    if w != 0:
        matriz_enlaces[:, j] = matriz_enlaces[:, j] / float(w)

#con el if se asegura de poder dividir la columna j por un valor no nulo.

```

De esta manera se obtiene la matriz de enlaces, donde si el aeropuerto `j` tiene como destino el aeropuerto `i` entonces se pisa en la posición `[i, j]` de la matriz de enlaces por 1 sobre la cantidad de enlaces externos del aeropuerto `j`, caso contrario se mantiene en 0.

La matriz de enlaces construida debe ser irreducible. Sin embargo, la dimensión de la matriz es de 3425 por 3425. Determinar si es irreducible es difícil de comprobar, por tener una dimensión muy grande y sería muy costoso computacionalmente. Por otro lado, se puede deducir que la

matriz construida no tiene valores negativos, solo posee valores positivos o iguales a cero. Entonces vamos a suponer que la matriz de enlaces es irreducible por construcción.

Por consiguiente, como la matriz de enlaces es irreducible se cumple la hipótesis del teorema de Perron-Frobenius, entonces deben existir un autovalor dominante y su correspondiente autovector, ambos positivos. Entonces, para poder encontrarlos se debe aplicar el método de las potencias. Antes de aplicar el método, se debe definir un vector inicial de la siguiente manera:

```
N = len(aeropuertos)
vector_inicial = (1/N) * np.ones(N)
```

El vector inicial se entiende como la probabilidad que tienen todos los aeropuertos en la primera iteración, que es la misma para todos $1/N$.

Recordando el método de las potencias:

```
def autpotencias(A, q_0, err=1e-10, M=500):
    q_tilde = A @ q_0
    rho_tilde = (q_0.T @ q_tilde)/np.linalg.norm(q_0, 2)**2
    for k in range(M):
        q = q_tilde/np.linalg.norm(q_tilde, 2)
        q_tilde = A @ q
        rho = q.T @ q_tilde
        if np.abs(rho - rho_tilde) < err:
            print("iteraciones:", k)
            return q, rho
        rho_tilde = rho
    return q, rho
```

Entonces, se procede en aplicar el método de las potencias, donde las entradas son, la `matriz_enlaces` construida y el `vector_inicial`. Por otro lado, las salidas son el vector `aeropuertos_rangos` y el `autovalor_proporcional`.

```
aeropuertos_rangos, autovalor_proporcional = autpotencias(matriz_enlaces,
vector_inicial)
```

El `autovalor_proporcional` indica la proporcionalidad entre la puntuación y el rango de los aeropuertos, y el vector `aeropuertos_rangos` son valores positivos que indican el rango de cada aeropuerto.

Ahora se debe encontrar los principales aeropuertos con mejor clasificación. Primero, se debe determinar los índices de ordenamiento de `aeropuertos_rangos` usando `np.argsort`, que los ordena de menor a mayor.

```
indices = np.argsort(aeropuertos_rangos)
```

Ya conociendo los índices de las clasificaciones se reordena las siguientes listas de acuerdo al orden de `indices`.

```
aeropuertos_rangos = aeropuertos_rangos[indices]
aeropuertos = aeropuertos[indices]
```

Pero lo que nos interesa es tener el orden de mayor clasificación a menor clasificación. Entonces se debe aplicar un `reversed()` para cambiar el orden de las listas.

```
aeropuertos_rangos = list(reversed(aeropuertos_rangos))
aeropuertos = list(reversed(aeropuertos))
```

Ya ordenados de mayor a menor las clasificaciones, se tomará los primeros 10 valores de rango y códigos para armar un top 10 de aeropuertos con mayor importancia.

```
top_diez_rangos = aeropuertos_rangos[:10]
top_diez = aeropuertos[:10]
```

Ahora por medios de los códigos se debe encontrar los datos de los 10 aeropuertos, para esto se usa `datos_aeropuertos`, donde la columna 4 tiene todos los códigos.

```
datos_codigos = datos_aeropuertos[:, 4]
```

Por la configuración que tiene `datos_aeropuertos` (cada elemento de la matriz se encuentra entre comillas), se debe eliminar las comillas de `datos_codigos` para evitar errores.

```
for i in range(len(datos_codigos)):
    datos_codigos[i] = datos_codigos[i].strip('")')
```

En el siguiente `for` se busca la posición del código en `datos_codigos` y, en base a esa posición encontrar la fila que le corresponde en `datos_aeropuertos` para obtener toda la información sobre el aeropuerto que esta vinculado con ese código. Por último, imprimir el top 10 de aeropuertos con mejor clasificación y, además se considera el último aeropuerto con la más baja clasificación.

```
print('Top 10 de los aeropuertos con mejor clasificacion en
el mundo de acuerdo a los datos de OpenFlights registrados en 2014.')
print('/')

print('Puesto /', 'Clasificacion /', ' Nombre /', ' Ciudad /',
      ' Pais /', ' Codigo /')

i = 0 #contador para los aeropuertos.
for k in range(len(top_diez)):
    #Se verifica si el codigo de 'top_diez' se encuentra en 'datos_codigos'.
    if top_diez[k] in datos_codigos:
        #Se busca la posicion del aeropuerto usando np.where.
        r = np.where(datos_codigos == top_diez[k])[0]
        i = i+1
        lista_ranking = datos_aeropuertos[r[0], 1:5]
        #Esta lista contiene la informacion sobre cada aeropuerto del top 10.

        for l in range(len(lista_ranking)): #Nuevamente se eliminan las comillas.
            lista_ranking[l] = lista_ranking[l].strip('")')

#Por ultimo, se imprime la informacion sobre los aeropuertos en orden decreciente.
#Por el mismo for se genera una tabla.

        print(f' {i} /', f'{top_diez_rangos[k]} /', f'{lista_ranking[0]} /',
              f'{lista_ranking[1]} /', f'{lista_ranking[2]} /', f'{lista_ranking[3]} /')
        print('-----')
        print('-----')

    else:
        print('Desconocido') #En caso que el codigo no exista.
```

```

#Aeropuerto con la menor clasificacion.

ultimo_areopuerto = aeropuertos[-1]

print('Por ultimo el aeropuerto con menor clasificacion es:')
print('---')
print('Puesto /', Clasificacion /', , Nombre /', , Ciudad /',
      Pais /', , Código /')
if ultimo_areopuerto in datos_codigos:
    r = np.where(datos_codigos == ultimo_areopuerto)[0]
    menor_ranking = datos_aeropuertos[r[0], 1:5]
    for l in range(len(menor_ranking)):
        menor_ranking[l] = menor_ranking[l].strip('')

    print(f'{len(aeropuertos)} /', f'{aeropuertos_rangos[-1]} /',
          f'{menor_ranking[0]} /', f'{menor_ranking[1]} /',
          f'{menor_ranking[2]} /', f'{menor_ranking[3]} /')
    print('-----')
else:
    print('Desconocido')

```

Finalmente, el Top 10 de los aeropuertos con mejor clasificación en el mundo de acuerdo a los datos de OpenFlights, se muestran en la siguiente tabla.

Puesto	Clasificación	Nombre	Ciudad	País	Código
1	0.15286348047817616	Frankfurt am Main Airport	Frankfurt	Germany	FRA
2	0.15033062074702938	Istanbul Airport	Istanbul	Turkey	IST
3	0.1498447170651457	Charles de Gaulle International Airport	Paris	France	CDG
4	0.14840692278730913	Amsterdam Airport Schiphol	Amsterdam	Netherlands	AMS
5	0.13811057624030693	Hartsfield Jackson Atlanta International Airport	Atlanta	United States	ATL
6	0.13260623857449214	Beijing Capital International Airport	Beijing	China	PEK
7	0.1291760432135052	Chicago O'Hare International Airport	Chicago	United States	ORD
8	0.12313806895225195	Domodedovo International Airport	Moscow	Russia	DME
9	0.12198150053731358	Munich Airport	Munich	Germany	MUC
10	0.11714838354587995	Dubai International Airport	Dubai	United Arab Emirates	DXB

Los aeropuertos obtenidos son de ciudades o países que son considerados importantes en el mundo, por ende sus aeropuertos poseen un gran volumen de tráfico de vuelos, entonces tiene sentido que estos aeropuertos estén entre los primeros lugares de clasificación. Desde el punto del método, los aeropuertos del top 10 tienen las más altas calificaciones debido a que existen enlaces de llegada también con buenas calificaciones. Por otro lado, el aeropuerto con menor clasificación es:

Puesto	Clasificación	Nombre	Ciudad	País	Código
3425	0.0	São Felix do Xingu Airport	Sao Felix do Xingu	Brazil	SXX

La clasificación del último aeropuerto es nula, uno de los principales causantes se debe a que el aeropuerto debe tener enlaces de llegada con muy baja clasificación. Si se desea ver del código de python se deja el siguiente enlace: [Proyecto Colab](#).

6. Conclusiones

Para finalizar con el informe se hará un análisis sobre los resultados obtenidos y, las ventajas y desventajas de haber aplicado el algoritmo de PageRank.

Como se pudo observar, el método de PageRank llegó a la solución deseada. El mismo encontró dentro de 3425 aeropuertos los que tienen las mejores clasificaciones y el aeropuerto con la menor clasificación. Desde un punto de vista general, la solución tiene sentido ya que, en la tabla de top 10 se encuentran las ciudades o países que en el mundo son importantes, debido a que tienen un alto volumen de tráfico de vuelos ya sea por turismo, negocios, trabajo, entre otros. Por otra parte, de la clasificación del top 10 se puede deducir que, los rangos de estos aeropuertos son altos ya que tienen más enlaces de llegada que de salida. Además, los aeropuertos con enlaces de llegada también tienen rangos altos. Esto se debe a que tienen un número limitado de enlaces externos. Por el contrario, en la última clasificación se puede observar que el rango es nulo. En este puesto se encuentra un aeropuerto de Brasil, donde su clasificación se vio afectada por qué tuvo más enlaces de salida que de llegada y también se debe a que los enlaces de llegada tampoco tienen una buena clasificación, por ende el método redujo su rango considerablemente hasta dejarlo en último lugar.

En cuanto al método de PageRank, una de las grandes virtudes que tiene es su capacidad de manejar grandes volúmenes de información, establecer una relación entre los aeropuertos de llegada y salida usando probabilidad, y encontrando de manera iterativa un vector de clasificación cuyas componentes son todas positivas. Este último se debe gracias al teorema de Perron-Frobenius, pues el teorema asegura que si la matriz es irreducible entonces existen un autovalor dominante y un autovector correspondiente, ambos positivos. Al método le interesa que las componentes del vector de Perron-Frobenius sean todas positivas para poder clasificar a los aeropuertos. En cuanto al autovalor dominante da la proporcionalidad entre los rangos y las puntuaciones de los aeropuertos. En el caso de aplicación, la matriz es irreducible por construcción y es por esto que se satisface el teorema de Perron-Frobenius.

Lamentablemente la matriz de enlaces que se armó para el método, no se pudo comprobar si era irreducible. Si bien en la parte teórica existen muchas formas de probar si una matriz es irreducible, en la práctica es difícil comprobar ya sea por su dimensión y/o el costo computacional. En el caso de la matriz de enlaces su dimensión era muy grande y probar que era una matriz primitiva iba a ser demasiado costoso. Sin embargo, era fácil deducir que la matriz construida era irreducible, pero esto no siempre sucede. Existirán casos donde será difícil deducir si una matriz es irreducible o no, entonces se deberá buscar métodos que permitan saber si una matriz es irreducible.

Para terminar, el método de PageRank supo ser algo muy novedoso en sus primeros años de existencia, ya que cambió drásticamente los motores de búsqueda. Con el tiempo el algoritmo se vio forzado a actualizarse debido a que se diseñaban métodos para manipular artificialmente el PageRank de una página, uno de los métodos más conocidos es el *spam* [6]. Actualmente, el algoritmo de PageRank dejó de actualizarse en 2013 y fue eliminado definitivamente de la Google en marzo de 2016. Sin embargo, Google sigue usándolo de forma interna y existen algunas herramientas que permiten conocer el PageRank de cualquier página.

Referencias

- [1] Airport, airline and route data, <https://openflights.org/data.html>.
- [2] The importance of Perron-Frobenius Theorem in ranking problems. Peretti, Alberto. Department of Economics, University of Verona.2014.

- [3] The Perron–Frobenius theorem and the ranking of football teams. Keener, James P.SIAM review 1993.
- [4] Damián Fernández, Autovalores y autovectores, <https://colab.research.google.com/drive/1aCMRbrRXpisjDQYI49rFdhtMvoMPqr0M?authuser=1>.
- [5] Teoría de Perron-Frobenius, <https://www.famaf.unc.edu.ar/documents/881/BMat47-2.pdf>.
- [6] PageRank, Wikipedia, <https://es.wikipedia.org/wiki/PageRank#:~:text=PageRank%20es%20una%20marca%20registrada,por%20un%20motor%20de%20b%C3%A1squeda..>