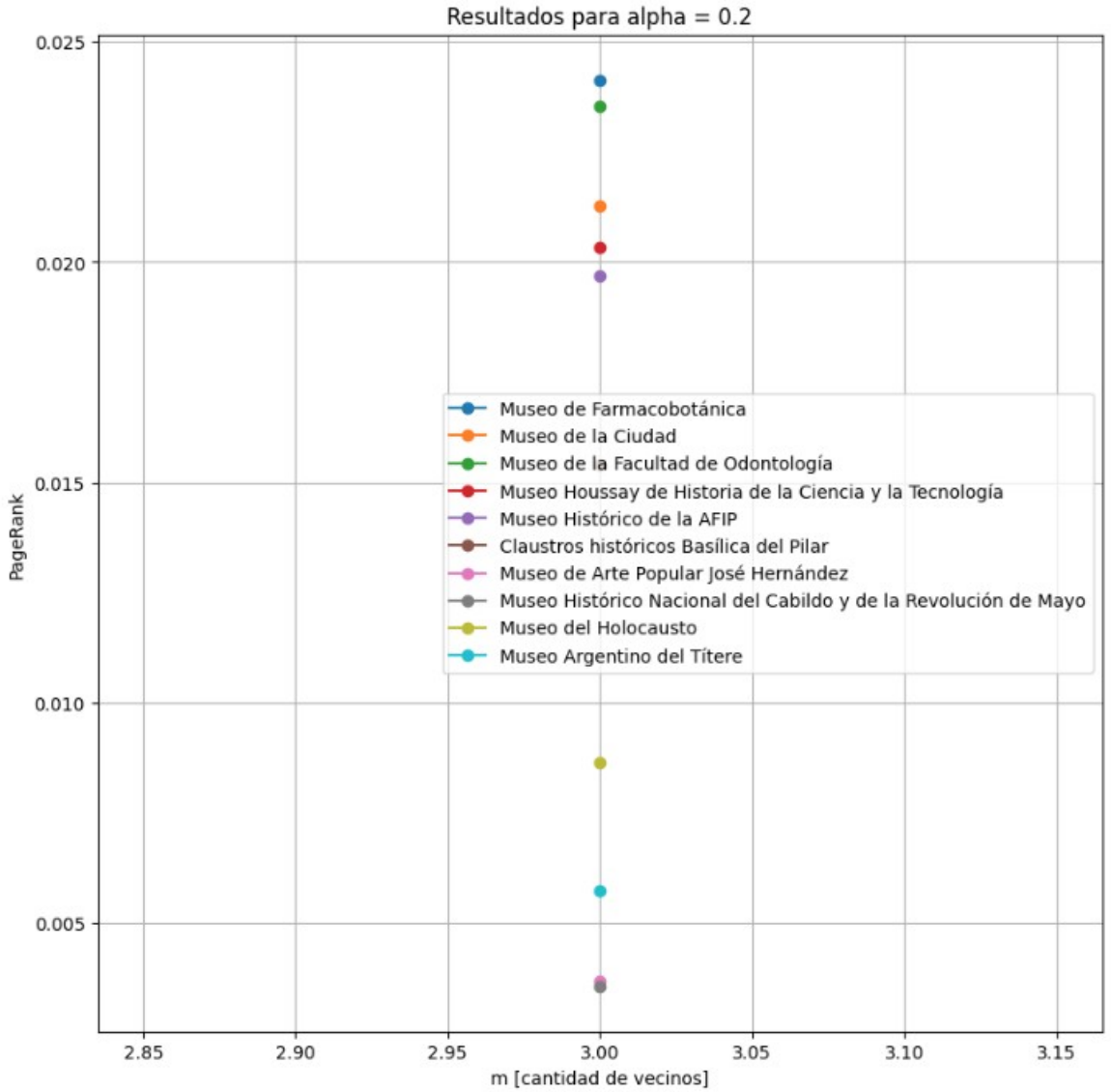
Correcciones TP1

Ej3. Bien-
Si esperaba que continuaran haciendo el barrido como hicieron en la primera entrega (pero no todos contra todos, dejando uno fijo y variando el otro como habiamos hablado). Aca por ejemplo.



Hubiese estado correcto mostrar en el eje x el alpha y ver como varían el ranking para distintos alpha (manteniendo m=3). Idem para barrido de m con alpha constante. Teniendo en cuenta que corrigieron lo esencial está OK.

Ej4

Bien
Detalle. Aqui

$$w_j = \sum_{i=1}^{n} \sum_{k=1}^{r-1} C_{ij}^{k} \cdot v_i$$

Creo que su Cij deberia ser Cji, ya que el indice a variar es j y se mueven en la fila. Idem para otros terminos.

Otro typo

Así, si v_i es el número de visitantes que arrancan en i, el número de visitas que llegan a j exactamente en el paso k es

$$\sum_{i=1}^{n} (C^k)_{ij} \cdot v_i.$$

¿No sería en el paso n?

Ej5

Bien

Ej6

Regular.
Añaden el análisis correspondiente pero introducen un nuevo error que no tienen en la entrega original! Esto refiere a cómo crean su cota.
En el enunciado del ejercicio menciona: *Supongan que se enteran de que el n´umero total de visitantes por museo w tiene un error del 5 %,*
¿Que quiere decir un error del porcentual de 5% si no es relativo a algo? Más precisamente en este caso es relativo a la magnitud de w.
Sin embargo ustedes hacen:

delta_normalizado = np.divide(norma_delta, norma_cant_visitantes)

Pero eso no tiene sentido porque lo que llaman norma_delta ya es la medida relativa. Esta division de más hace que la cota del error relativo de v les quede mucho más pequeña de lo que en realidad es.

Correcciones TP2

Ej1

a) L. Bien
Q. Bien
Entiendo que en el paso de la primera ecuación a la segunda aca hay un typo. El 2E debería estar dividiendo.

$$\sum_{j=1}^{n} A_{kj} = \lambda_2 + \sum_{j=1}^{n} \frac{k_k \cdot k_j}{2E}$$

$$\sum_{j=1}^{n} A_{kj} = \lambda_2 + k_k \cdot \sum_{j=1}^{n} k_j 2E$$

b) Bien
c) Bien

Ej2

a) Bien
b) Bien
c) Bien

Ej3

a) Mal. Tambien invalida todo lo que desprenda de modularidad. Cosa que también tendrán que revisar y re-escribir cuando tengan sus nuevos resultados.
Aca

```
def calcula_P(A: np.ndarray) -> np.ndarray:
    """
    Construye la matriz P que cumple:
    Pij = k_i * k_j /2E
    """
    k = np.sum(A, axis=1)  # <-- grados
    suma_doble_E = np.sum(k) / 2  # <- constante 2E
    P = np.outer(k, k) / suma_doble_E
    return P
```

Si k tiene los grados, entonces la np,sum(k) tiene 2E. ¿Para que dividen por 2? Esto hace que esten dividiendo ki*kj/E. En lugar de 2E. Esto es importante porque cuando construyen R, esto es equivalente a A - 2P en lugar de A - P. Es **OTRA matriz.** Con lo cual invalida todo lo que continua con modularidad. Pueden ver por ejemplo que si a su implementación de la matriz R la multiplican por el vector de 1s para el grafo de prueba les da [-3, -3…..] en lugar del vector nulo

Notar que este problema también lo pueden evidenciar cuando piden el autovector de autovalor dominante, Les da el vector de 1s. Si eso estuviera bien significa que el dominante tiene autovalor 0 (porque como ya mostraron, el autovalor asociado a este es 0). Entonces todos son 0s y R debería ser la matriz nula!

Muy lindos diagramas
b) Bien-
Detalle. En metpotI2 no eliminan el mu al momento de devolver el autovalor.
Tambien estaría bueno, que se desarrollen un poquito con respecto a la particion esperada. ¿Se asemeja? ¿Es muy diferente?
c) Regular.
Falta el analisis con el grafo de ejemplo pedido
Además, Con respecto a modularidad_iterativo. No me esta quedando claro algunos detalles del algoritmo. Me gustaria, si consideran que es correcto que agreguen una explicación detallada justificando porque funciona y cual es su idea. En particular ustedes hacen mejor_delta_Q=0. ¿Es cierto que siempre que exista un corte con modularidad positiva conviene cortar a no cortar? En otras palabras, ¿No podria darse que no cortar ninguna comunidad sea mejor para la modularidad que cortar alguna, incluso con modularidad positiva? No se esta respuesta pero su algoritmo si entiendo que su algoritmo toma ciertas propiedades para funcionar que no me resultan del todo obvias. Recomiendo explicitarlas y porque se dan o rehacer el código.

Ej4

Tiene los problemas mencionados más arriba para modularidad, lo cual invalida los resultados y análisis (y dado que los niveles del laplaciano también dependen de este, idem). Corregir.
Más allá de esto, bien el codigo.

Ej5

Bien. Revisar y cambiar los elementos mencionados mas arriba como su comentario respecto al error en el TP1.