Organización de Datos 75.06. Primer Cuatrimestre de 2018. Examen parcial, segunda oportunidad: Resolucion

1- La parte más compleja del primer punto es encontrar los hashtags que fueron publicados únicamente por usuarios de Argentina (para encontrar los hashtags antes que nada hay que aplicar algún flatMap para splittear por palabras y luego quedarse con las palabras que comienzan con #). Si hacen un join con usuarios de Argentina y se quedan con esos hashtags está mal. Para poder encontrar cuáles son los hashtags publicados únicamente por usuarios de Argentina, se puede realizar un inner join entre las publicaciones y los usuarios que NO son de Argentina, y con eso se puede obtener un listado de los hashtags que no se quieren. Con lo cual los hashtags que necesitamos son todos los que están en publicaciones pero que no pertenecen al listado que obtuvimos anteriormente. Finalmente, para resolver este punto habrá entonces que contar la cantidad de hashtags, ordenarlos y tomar los primeros 10.

Otra alternativa: Se puede realizar un Left Join entre publicaciones y usuarios de Argentina, y quedará algo de la siguiente forma:

Hashtag	País	Id Usuario
Starwars	AR	1
Starwars	-	-
Starwars	AR	2
Datos	AR	2
Datos	AR	3
Datos	AR	4
FIUBA	-	-
FIUBA	-	-
FIUBA	AR	5

En la tabla anterior podemos ver que los registros nulos surgen porque hay usuarios que no son de Argentina. En este caso nos interesa únicamente obtener el hashtag "Datos". Podemos ver que el Hashtag "Datos" apareció 3 veces y además tiene 3 registros no nulos (esto nos sugiere que podemos llevar dos contadores: uno para la cantidad de Hashtags y otro para contar la cantidad de registros no nulos que hay. Luego podemos verificamos que ambos contadores sean iguales y esos serán los hashtags que queremos).

Para resolver el punto 2, se puede obtener un RDD de la forma (id_usuario, id_hashtag) y hacer el inner join contra sí mismo a través de id_hashtag. Con eso ya podemos obtener todos los pares que necesitamos. Es necesario, una vez generado el RDD, hacer un .cache().

Map innecesarios descuento de 3 puntos. Join sin claves compatibles descuento de 5 puntos. No filtrar lo antes posible descuento de al menos 3 puntos. No usar el método cache en el punto 2 descuenta 3 puntos.

2- Por un lado es necesario hallar la máxima cantidad de seguidores que puede tener un usuario en Argentina. Luego es necesario filtrar aquellos usuarios que son de argentina y tienen mas de 80% de seguidores de ese valor maximo. Existen distintas formas de hacerlo, por ejemplo se puede generar esos valores por columna y evaluar contra la otra columna la condicion de filtro o directamente calcular el 80 como condicion a realizar. No hay penalizaciones si filtran por un valor calculado en una variable externa y no en una columna del dataframe.

Por otro lado se debe realizar tambien un filtro considerando que cantidad_seguidos de seguidos para ese usuario no supere el 15% de la cantidad de seguidores que tiene el usuario.

Una vez seleccionados los candidatos generales se debe obtener el top 10 considerando, la cantidad de seguidores.

Si no realizan filtros por pais, descuento de 10 puntos.
Si para obtener el top 10 no usan nlargest descuento de 5 puntos (por ejemplo si usan sort_values).
No se evalua como mejor si hace una solucion generica para todos los paises, dado que el top 10 lo queremos para Argentina.

3- V/F
a- En T-SNE, si dos puntos A y B a distancia 1 se consideran como cercanos entonces si C y D (distintos a A y B) también están a distancia 1 también se consideran cercanos.
Este es un poco complicado. La respuesta es falsa, porque cada punto tiene su propio parámetro Sigma que regula el radio del gaussiano:

$$\exp(-||x_i - x_j||^2 / 2\sigma_i^2)$$

Si por ejemplo C y D tuvieran parámetros Sigma mucho más chicos que los de A y B entonces no se podrán considerar tan similares como A y B.
b- Para matrices grandes la SVD no es una opción para reducir dimensiones por un tema de eficiencia.
La respuesta también es falso. Hemos hablado de esto en clase, y con métodos iterativos se pueden obtener los K autovectores que se necesiten (algoritmo de Lanczos). Se puede programar en apache spark.
c- Falso. En PCA, se obtendrá una mayor varianza en los datos si los proyectamos contra la primer componente principal.
d- Dado que t-SNE escala cuadráticamente dada la cantidad de inputs que tenga no podemos utilizarlo para grandes volúmenes de datos.
Esto es falso dado que es posible utilizar alguna otra técnica anteriormente como truncatedSVD o PCA para reducir la cantidad de dimensiones a un número manejable y usarlo.

4- La tabla queda de la siguiente forma:

	v1	v2	v3
r1	-1	-1	1
r2	1	-1	1
r3	1	-1	1
r4	-1	-1	-1
r5	1	-1	1
r6	1	-1	1

La familia de funciones de hashing tendrá la forma h(x1,x2,x3)=(ax1+bx2+cx3+d) mod 17 mod 13 (el número 13 es por enunciado, el número 17 es porque se exige un número primo mayor o igual a 13). Se debe cumplir lo siguiente:

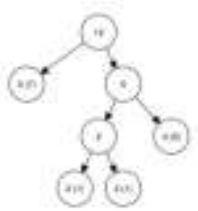
- h(-1,1,1) = h(-1,-1,-1) y a su vez distinto a h(1,1,1) porque solo deben colisionar v1 y v2
- h(-1,1,1) = h(-1,-1,-1) porque en la segunda banda tienen que colisionar todos

El punto es entender cómo encontrar a,b y c para cumplir con el enunciado. De la primer restricción, para que h(-1,1,1) = h(-1,-1,-1) debe ocurrir que -a+b+c+d = -a-b-c+d en mod 17 mod 13. Si se asume que a y d pueden ser cualquier valor, entonces nos podemos centrar en que b+c = -b-c (en mod 17 mod 13). Con b=c=1 vemos que se cumple lo pedido. Podemos asignar algún valor a la variable a tal como a = 1 (y d=0) y se puede verificar que -1+1+1+0 mod 17 mod 13 es igual a 1 mientras que -1-1-1+0 mod 17 mod 13 = 1 con lo cual cumple con que v1 y v2 colisionan. Para verificar que v3 no colisione en esa banda, verificamos que 1+1+1+0 mod 17 mod 13 = 3 que es distinto a 1.
En la segunda banda, se puede verificar que colisionan todos porque en la segunda banda los valores para v1 y v2 son los mismos que en la primer banda. A su vez, v1 es igual a v3 con lo cual colisionará

5- Deben partir armando el índice invertido con FC parcial (n=3). Si arman mal la estructura del índice, no concatenan los términos y los punteros, el ejercicio vale cero. Luego, sobre la estructura armada deben explicar cómo se realiza una búsqueda binaria sobre el índice, de cada uno de los dos términos. Si hacen busqueda secuencial vale cero ya que carece de sentido tener un índice. Deben contar los accesos al índice, y luego los accesos al léxico para reconstruir cada término, y por último una vez encontrado cada uno de los dos términos buscados acceder a los punteros a documentos.

6- El algoritmo de block sorting en este ejercicio consiste de tres etapas:
Transformación de BW
Move to front
Huffman estático (se deduce que es estático por la tabla de frecuencias)

Para descomprimir, hay que armar todo el camino inverso. Entonces primero es necesario armar el árbol de Huffman siendo que queda el siguiente:



Con este árbol vemos que la codificación es la siguiente:
Símbolo 0: 0
Símbolo 2: 100
Símbolo 3: 101
Símbolo 4: 11

Entonces 1111010000101001100 será 440200300400. Con este resultado ahora podemos revertir el MTF usando el vector [ABCDE]:
4 = E [EABCD]
4 = D [DEABC]
0 = D [DEABC]
2 = A [ADEBC]
0 = A [ADEBC]
0 = A [ADEBC]
3 = B [BADEC]
0 = B [BADEC]
0 = B [BADEC]
4 = C [CBADE]
0 = C [CBADE]
0 = C [CBADE]
=> EDDAAABBBCCC

Ahora tenemos que revertir la transformación de BW y, para ello, tenemos que utilizar el índice L = 1.

3A0
4A1
5A2
6B3
7B4
8B5
9C6
10C7
11C8
1D9
2D10
0E11

Como L=1, la primer letra del archivo es A, y A apunta al registro 4 que es B, y B apunta al registro 7 que es C, y así sucesivamente. El archivo final será:

ABCDABCEABCD

7- Se evaluara la creatvidad propuesta para la solucion. Existen diversas cosas que se podrian plantear, algunas que funcionen por ejemplo de forma interactiva, por ejemplo:

- Comparar la distribucion de los horarios o dias en los que utilizan ciertos hashtags entre si.
- Comparar la cantidad de usos de hashtags en cantidad de publicaciones por pais para un conjunto de hashtags
- Wordcloud utilizando los hashtags dandole el peso de cantidad e publicaciones que los utilizan.
- Una estructura de grafo utilizando como nodos los usuarios y conexiones entre usuarios los distintos hashtags o viceversa