TRABAJO PRACTICO N°2 – SPARK

ORGANIZACIÓN DE DATOS 75.06

Alumno: Gonzalez, Agustin Nicolas

Padrón: 106086

Corrector: Matias.

Link al colab:

https://colab.research.google.com/drive/1JSeQLfMFR-CtBhXqrr0kcy7ZWFdvZL-N?usp=sharing

```
[ ] #1
    #Considerando los logs de acciones realizadas sobre ítems, mostrar el top 10 de ids
    #de ítems que fueron afectados por mayor cantidad usuarios distintos
    sqlContext = SQLContext(sc)
    df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/logs.csv', header=True, inferSchema=True)
    rdd = df.rdd

    rdd.filter(lambda x: x.title != None).map(lambda x: (x.title, 1)).reduceByKey(lambda x,y: x+y).takeOrdered(10, lambda x: -x[1])
```

```
[('Especial:Userlogin', 75423),
 ('delete', 13761),
 ('move', 5972),
 ('Usuario:Ontzak', 4993),
 ('create', 4940),
 ('Usuario:Marcelo', 4383),
 ('Usuario:Taichi', 2807),
 ('Usuario:Geom', 2805),
 ('Usuario:Fixertool', 2788),
 ('Usuario:Strakhov', 2705)]
```

```
#34
#¿Quién es el usuario que más ha bloqueado a otros?
sqlContext = SQLContext(sc)
df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/logs.csv', header=True, inferSchema=True)
rdd = df.rdd

rdd.filter(lambda x: x.action=="block").map(lambda x: (x.contributor_username, 1)).reduceByKey(lambda x,y: x+y).reduce(lambda x,y: x if x[1] > y[1] else y)
```

```
('Magister Mathematicae', 29226)
```

```
[ ] #14
    #El Top 5 de contenidos que tienen la mayor cantidad de redirecciones que apuntan a
    #ellos
    sqlContext = SQLContext(sc)
    df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/redirect_list.csv', header=True, inferSchema=True)
    rdd = df.rdd

    rdd.map(lambda x: (x.rd_title, 1)).reduceByKey(lambda x,y: x+y).takeOrdered(5, lambda x: -x[1])
```

```
/usr/local/lib/python3.7/dist-packages/pyspark/sql/context.py:79: FutureWarning: Deprecated in 3.0.0. Use SparkSessic
  FutureWarning
[('Artículo_futuro', 1161),
 ('IV_milenio', 1001),
 ('V_milenio', 998),
 ('Estaciones_de_Metrobús_de_la_Ciudad_de_México', 396),
 ('163.117.0.0', 310)]
```

```
[ ] #32
    #¿Cuál es el segundo contenido con más referencias geográficas asignadas?
    sqlContext = SQLContext(sc)
    df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/geo_tags.csv', header=True, inferSchema=True)
    rdd = df.rdd

    #sqlContext = SQLContext(sc)
    #df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/contents', header=True, inferSchema=True)
    #rdd_contenidos = df.rdd

    rdd.map(lambda x: (x.gt_page_id, 1)).reduceByKey(lambda x,y: x+y).takeOrdered(2, lambda x: -x[1])[1][0]
    #rdd_contenidos.filter(lambda x: x.id == id_contenido).collect()
```

```
FutureWarning
7421943
```

Aclaracion sobre el 32: El resultado que se muestra es el segundo ID con mas referencias geográficas asignadas, lo que esta comentado era lo que había realizado en primera instancia, que era asignar ese numero como "id_contenido" y luego encontrar en el rdd_contenidos cual era el contenido con ese id, pero lo comente porque no encontraba nada

```
#20
Cantidad de Stubs por categoría en la Wikipedia.
sqlContext = SQLContext(sc)
df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/categorylinks.csv', header=True, inferSchema=True)
rdd = df.rdd

texts = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/contents_text_sample.csv')
texts.to_parquet('contents_text_sample.parquet')
del texts
df = sqlContext.read.parquet('contents_text_sample.parquet')
rdd_contenidos = df.rdd

stubs = rdd_contenidos.filter(lambda x: x.text != None and "url" not in x.text and len(x.text) < 1500).map(lambda x: (str(x.id), x.text))
rdd_filtrado = rdd.map(lambda x: (x.cl_from, x.cl_to)).join(stubs)
rdd_filtrado.map(lambda x: (x[1][0], 1)).reduceByKey(lambda x,y: x+y).collect()
```

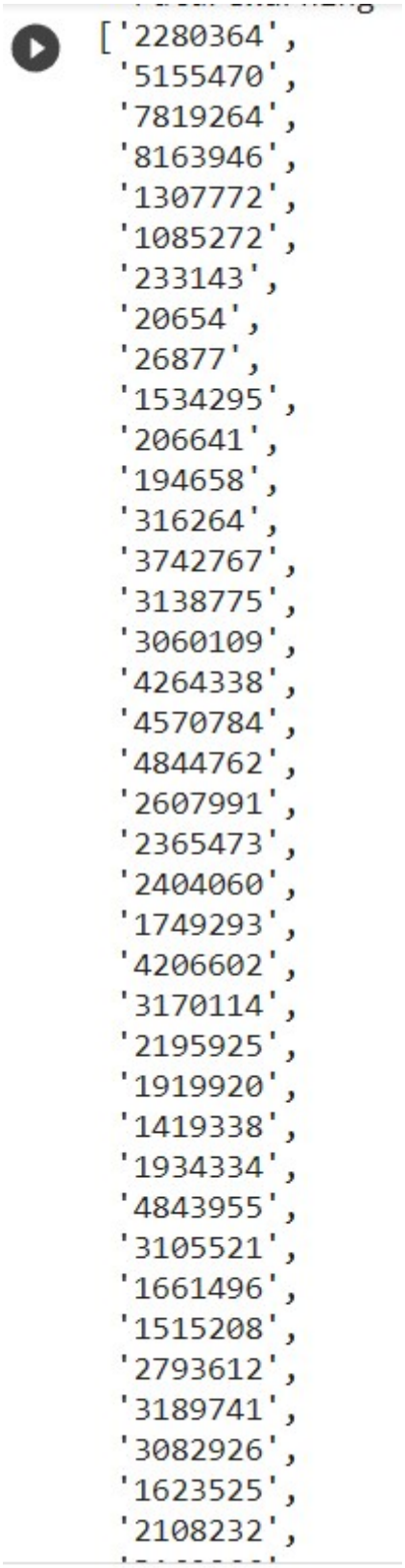
```
[('Wikipedia:ArtÃ\xadculos_con_identificadores_Microsoft_Academic', 423),
 ('AÃ±os_1210', 1),
 ('Armas_de_las_artes_marciales_de_JapÃ³n', 2),
 ('FÃ\xadsica_de_la_materia_condensada', 1),
 ('Poetas', 1),
 ('TerminologÃ\xada_arqueolÃ³gica', 2),
 ('Emiratos_Ã\x81rabes_Unidos', 2),
 ('Lenguaje_y_comunicaciÃ³n_sonora', 1),
 ('Localidades_de_RÃ\xado_Grande_del_Sur', 17),
 ('Portal:Colombia', 1),
 ('Nacidos_en_1951', 9),
 ('Wikipedia:ArtÃ\xadculos_con_identificadores_Open_Library', 87),
 ('Provincia_de_Fermo', 1),
 ('Wikipedia:CategorÃ\xadas_aÃ±adidas_por_plantillas', 48),
 ('Deportistas_de_Cantabria', 2),
 ('Distritos_de_Helsinki', 2),
 ('CientÃ\xadficos_de_Escocia', 4),
 ('Viola', 1),
 ('Wikipedia:Consultas_de_borrado_con_resultado_por_determinar', 25),
 ('Ã\x81lbumes_de_rap', 5),
 ('Jardines_botÃ¡nicos_por_paÃ\xads', 2),
 ('Finales', 1),
 ('Ã\x93pera_de_CanadÃ¡', 1),
 ('Terroristas_de_Colombia', 1),
 ('Siglas', 46),
 ('Transporte_de_la_UniÃ³n_Europea_por_paÃ\xads', 1),
 ('Estados_y_territorios_fundados_en_1802', 1),
 ('Reyes_de_Arda', 1),
 ('Fallecidos_en_los_aÃ±os_1480', 2),
 ('Infraestructuras_de_los_PaÃ\xadses_Bajos', 3),
 ('IntÃ©rpretes_de_mÃºsica_clÃ¡sica_de_EspaÃ±a', 2),
 ('Wikipedia:ArtÃ\xadculos_con_identificadores_PersÃ©e', 7),
 ('Pandaceae', 1),
 ('Localidades_de_Belice', 2),
 ('Novelas_en_inglÃ©s', 8),
 ('Premio_Locus_a_la_mejor_novela_de_fantasÃ\xada', 1),
 ('520', 1),
 ('MÃºsica_clÃ¡sica_por_paÃ\xads', 5),
```

Aclaracion sobre el 20: En primera instancia que el resultado continua, pero solo muestro esta parte, y luego es sobre el criterio que use para determinar cuando un articulo es un stub, tome como referencia lo que decía en la pagina de Wikipedia, menos de 1500 caracteres, y que no sea links a otras paginas, por eso también bloquie la palabra “url” que observe que se presentaba en muchos casos que eran simplemente para contener el link

```
#8
#Considerando el pagelink_sample.csv, usando una representación de grafos obtener
#aquellos contenidos que tienen “relaciones no correspondidas”. Entendemos como
#funciona una relación correspondida con un ejemplo: Si el contenido A tiene un link
#al B, pero B no tiene un link a A, podemos decir que B tiene una relación no
#correspondida con A
sqlContext = SQLContext(sc)
df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/pagelinks_sample.csv', header=True, inferSchema=True)
rdd = df.rdd

sqlContext = SQLContext(sc)
df = sqlContext.read.csv('/content/drive/MyDrive/Colab Notebooks/contents.csv', header=True, inferSchema=True)
rdd_contenidos = df.rdd

union = rdd.map(lambda x: (x.pl_title, x.pl_from)).join(rdd_contenidos.map(lambda x: (x.title, x.id))).map(lambda x: (str(x[1][0]), str(x[1][1]))).cache()
relaciones = union.map(lambda x:tuple(sorted(x))).map(lambda x:(x,1)).reduceByKey(lambda x,y: x+y).filter(lambda x:x[1]==1).map(lambda x: (x[0][0], x[0][1]))
relaciones_filtradas = relaciones.map(lambda x: (x[0], 1)).reduceByKey(lambda x,y: x+y) + relaciones.map(lambda x: (x[1], 1)).reduceByKey(lambda x,y: x+y)
relaciones_filtradas.map(lambda x: (x[0])).collect()
```



Aquí también el resultado continua.