Tarea 3

Andrés Abbott

November 16, 2017

1 Introducción

En el siguiente informe, se explicara el procedimiento utilizado para graficar la puntuación otorgada por los usuarios de Yelp según cada categoría. En primer lugar, se mostrara el procedimiento realizado para captar los datos de los archivos de Yelp, para posteriormente, mostrar el procedimiento realizado para graficar dichos datos. Las pruebas correspondientes se llevaron a cabo con 10.000 y 30.000 datos.

2 Captura de datos

El primer paso, fue crear los archivos csv para graficar. Para esto se utilizo la libraría Mrjobs de python. El script consiste en realizar un join entre dos archivos csv preexistentes de la base de datos de yelp. Esto dos archivos son reviews.csv y bussiness.csv.

En el mapper, lo que se hizo fue ir recorriendo los reviews e ir guardando los ids de los bussiness junto con la puntuación que este tuvo. Posteriormente, se recorrió el archivo bussiness para ir comparando los ids guardados anteriormente. Una vez coinciden los ids, se procede a buscar las categorías de este, retornando el nombre de la categoría y la cantidad de estrellas que tuvo.

Listing 1: Mapper

```
def mapper_userid(self, _, line):
  aux.append([line['business_id'], line['stars']])
  if 'categories' in line:
    for rev in aux:
      if rev[0] == line['business_id']:
          for cat in line['categories']:
            yield [cat, rev[1]]
```

En el reducer, lo que se hizo fue ir agrupando la cantidad de estrellas resididas por cada categoría, para así, poder tener todas las estrellas otorgadas unidas en una sola categoría.

Listing 2: Reducer

```
def reducer(self, key, values):
            aux_gen = list(values)
            yield [key, aux_gen]
```

Por ultimo, para crear los archivos csv, se utilizo se capto el output del script una vez que termino de correr. Par esto se utilizo el comando : python tarea3.py > out.csv

Cabe destacar que dicho procedimiento se pudo haber paralelizado. Si los archivos contienen un gran volumen de datos, se podrían dividir en pequeños sub archivos para facilitar el procesamiento. El único problema es que al momento de realizar los joins, se tiene que asegurar que los bussiness.id de los reviews estén dentro del archivo bussiness. Para esto, se puede realizar un preprocesado para ir ordenando los datos y luego dividirlos. La otra opción seria solo dividir solo el archivo reviews, pero eso generaría que se tenga que mantener una copia del archivo bussiness por cada sub archivo review.

Figure 1: Datos crudos.

3 Graficar

Para graficar, utilizamos el sistema Jupyter, que nos permite correr archivos en servidores remotos. Esto se utiliza para paralelizar y dejar el procesado de datos en servidores externos. La librería que se utilizo para graficar los datos fue Lightning de python.

En primer lugar, se procedió a leer el único csv generado anteriormente. Dicho csv tiene por cada linea el nombre de la categoría y la cantidad de estrellas.

Listing 3: Ejemplo csv

```
categoria_1  [2,2,3,4,5,5]
categpria_2  [2,3,2]
categoria_3  [5,3,4,3]
```

Al momento de ir leyendo las categorías, se tuvo que hacer una limpieza de toda la linea asociada a la categoría para extraer caracteres que impidieran convertir la lista de valoraciones en un arreglo. Un ejemplo de esto es la interpretación que le pueda dar la librería csv al carácter de termino. Al momento de abrir el archivo con un editor de texto normal, podemos apreciar que todo se ve como se escribió originalmente, pero al momento de leer la linea con la librería csv de python, notamos que todas las lineas tienen el conjunto de símbolos:

```
\xef\xbb\xbf
```

Otros arreglos que tuvieron que hacerse fue el de separar los términos de la lista, extraer comillas y agregar elementos faltantes para poder crear una lista de python para posteriormente poder convertir la lista en un numpy.array, que es el formato que recibe la librería Lightning para graficar. Un ejemplo de los datos crudos se puede apreciar en la figura 1

Cabe destacar que para llevar un mejor procesamiento de los datos e intentar paralelizar al momento de graficar, se puede separar cada categoría en un csv individual, para que así, cada instancia de Jupyter se preocupe de graficar solo una categoría.

4 Resultados

Debido a que existen mas de 500 categorías, no se analizaron a fondo todos los resultados, no obstante, como se menciono anteriormente, si se quisiese hacer, bastaría con correr nuevas instancias de Jupyter por cada categoría.

En cuanto al rendimiento, al momento de analizar los 10.000 datos y los 30.000 no se encontró gran diferencia al momento de graficar. Esto se debió a que solo se analizaron los graficos de algunas categorías, no obstante, puede ser que al momento de analizar todos los graficos, nos encontremos con categorías con una cantidad de valoraciones muy superior a las demás, por lo que el tiempo al momento de graficar podría variar.

En donde si vario el tiempo fue al momento de limpiar los datos y generar lo archivos csv. Al correr el script para generar los csv con los 10.000 datos, este se demoraba alrededor de 4 segundos, mientras que con 30.000 el tiempo podía llegar a subir a los 10 segundos o mas. Por otro lado, cuando se corrió el script para limpiar los datos, notamos que tanto con 10.000, como con 30.000 se tuvieron tiempos similares. Solo hubo una diferencia en unos segundos.

En cuanto a las categorías revisadas, no se entrono ningún patrón que nos pudiese ayudar a llegar a alguna conclusión útil, como tipo de distribución o patrón de datos.