Entrega 2

Andrés Abbott

October 10, 2017

1 P1

1.1 Review más único

1.1.1 Dessarrollo

En esta pregunta, se procedió a calcular el review mas único, para esto, se analizo cada palabra de los reviews y se reviso la cantidad de veces que esta se repetía en el review.

En la primera parte, la función del mapper es crear un mapa que contenga como clave una palabra y el review id asociado a esta. El valor que le pondremos a esta sera un uno para poder contar las palabras que repiten en los reviews.

```
def mapper(self, _, line):
        review_id = line['review_id']
        words = line['text'].split()
        for word in words:
                yield [word.lower(),review_id], 1

# review = entrega dos bigdata entrega
# local_id = 4
# mapper = [entrega, 4], 1
# mapper = [dos, 4], 1
# mapper = [bigdata, 4], 1
# mapper = [entrega, 4], 1
```

El reducer que acompaña al maper es el encargado de contar las apariciones de las palabras en el review. La llave que retorna sera la misma que en el maper anterior.

```
def reducer_total_words(self, key, count):
        yield key, sum(count)
```

Tomando el ejemplo anterior,

```
# reducer = [entrega, 4], 2
# reducer = [dos, 4], 1
# reducer = [bigdata, 4], 1
```

Los siguientes reducer realizaran algo parecido que el anterior, solo que esta vez, se encargara de contar las palabras únicas dentro del review. En este caso, las que solo aparecen una sola vez. Cuando se terminan de contar, se agrega la key 'max' para poder sacar el numero máximo de palabras únicas de todos los reviews.

```
def reducer_count_words(self, word_with_rev, count):
        aux = list(count)
        if int(aux[0]) < 2:
                yield word_with_rev[1], 1
def reducer_agroup_words_by_rev(self, rev_id, count):
        yield 'max', [sum(count), rev_id]


# reducer_anterior = [entrega, 4], 2 -> no se contara
# reducer_anterior = [dos, 4], 1
# reducer_anterior = [bigdata, 4], 1

# local_id = 4
#reducer = max , [2,4]

def reducer_max_word(self, rev_id, count):
        yield rev_id, max(count)
```

1.1.2 Resultados

Cuando se probo con el set de datos de 10.000, arrojo un timpo de 37.67 segundos, y el review mas unico correspondia al IMGMeUiadh0Ku1p-aajBJg Si revisamos el texto de este notamos que es el texto mas largo, por lo que tiene sentido el resultado:

text":"There are so many things *right* with Hana that one can almost overlook the sordid quality and schmaltzy presentation of their sushi rolls.nOverriding every other concept of the restaurant is the fact that 95% of the hungry patrons at Hana are boozehounds like myself, and this place will cater to the unholy amounts of BYOB that each table will bring. What is truly exceptional is that 1) No corkage fee 2) They will chill your white for you in an ice bucket at the table 3) If you bring beer, they will keep it in their fridge for you and bring you another as you continue to dine 4) everyone here relishes in this policy.Seriously, I witnessed a table of 5 with 5 bottles of wine and a sixer of some imported swill. You could eat a seaweed wrapped piece of Gypsy at this saturation point, and it would taste like toro.nSo, my gf Melissa B and I get there around 6:30 on a Saturday loaded up with our Sav Blanc and six of Rogue Dead Guy Ale and with full intentions of waiting for a table due to the popularity of this restaurant and its 50 person max capacity. Surprisingly, they had 3 tables open, all of them 4-tops. The hostess informs us that the wait will be 20-25 mins and I'm thinking 'oh, cool, those tables must be reserved. No worries. Oh, and hey look! There is a two top paying out'. Well, after that couple leaves, they scoot over the two top to make it a six top and seat the aforementioned boozer-quintuplets who were rightfully there before us still waiting. Okay. No biggie, there are still open tables.A party of three come in. "Hi! There is no waiting". They get a seatAnother party of three stroll in. Yep. "Hi! There is no waiting". There goes our last chance of a seat for a while, and we're 25 mins into our wait. Thank God I brought the beer.As I understand it, they will only seat parties of two at their two tops [of which they had few, and one was being used as an impromptu wine rack for the wino-Salingers] or at the sushi bar. I really have absolutely no problem with this policy, but I could see how it would piss some people off – maybe from an older generation who feel slighted by shit like this, which is why I'm even discussing it. But anyway, we eventually we got our perfect little two top and preceded to order edamame and miso. These are two staple appetizers for any Japanese restaurant. The edamame came out cold, which was okay, as it is typically served cooled .. but I actually prefer my edamame heated and the server never asked for a preference ... hell, they even ask you at throw-away joints like Ra. Since we are a culture of benevolent Americans who are intent on skewing EVERY foreign dish, I always assume that the edamame will be served warm. Either way, I was too timid to ask since these servers seem to be flying around the small restaurant with nary a second to spare at the microwave. The miso soup was excellent ... but if I'm looking for shit to complain about, they served it with that damn goofy, oversized spoon.When we got our boat of fish my gf and I commented that we would have liked a little extra prep time. I mean come on, there needs to be distinct transitional times between courses. We were only mildly impressed with the presentation. The rolls were prepped very sloppily. The yellowtail was absolutely lost in the rice. The tempura shrimp was sliced so thin that it was destroyed by the chopstick. The caterpillar roll had almost no essence of eel within. The spicy tuna was good, but nothing to rave about as seems to be the trend when discussing Hana. We also ordered the processed daikon off of their specials-board [which our server never went over BTW], and I would highly recommend staying away from this unless you curiously want to know what Little Lisa's Slurry from the Simpson's episode "The Old Man and The Lisa" might have tasted like. Blech. I'm sure it was downhome authentic, but Christ, THAT was vile. Anyway, that was MY fault, not Hana's. will say that the first time that I came to Hana [with Graham C and Yelp's famous Jerry], it was absolutely phenomenal. I think something has been lost in translation. Back then, their rolls were much better prepared and the toro was otherworldly. Perhaps it is the hulking Melrose virulence seeping into their foundation.I will conclude with a statement about how I am aware that Hana is going for authenticity; which makes the curt service, lack of formalities like hi! how is everyone today" [I doubt they do that shit at sushi joints in Japan], and loud, bustling atmosphere work to a degree. I

can HOWEVER, see that being such a turn-off to those unwilling or unaware to put up with it, or worse yet, interpret it as rudeness. I personally like the fact that places like Hana exist as an antithesis to garbage like Ra. But every polarization needs an equilibrium, and for my money, time, and need for experience, nothing beats

Al momento de hacerlo con el set de 30.000datos, arrojo un tiempo de 1minuto 57 segundos. EL texto que resulto ser mas unico esta ves es el con el id VQsjcpThYG9Pap-5n11OX que corresponde al texto

text":"The Atlantic, on Dundas west of Dufferin, is unassuming from the outside. The plastic sign, with its generic lettering and palm trees and sailboat all blue on a white background, is a camouflage, blending in with the numerous sports bars, cafes, and family restaurants that line this strip of Little Portugal. The interior, dark and wooden, is punctuated with decor that hints at the individuality and sophistication of the cuisine, while retaining elements of kitsch; on the wall behind our table, a small penciled drawing of a menacing thresher shark is bordered by two large, softly-coloured ballerinas. The cheerfully cluttered bar dominates the back half of the room, with a view into the tiny kitchen and a large poster on the wall: Good Things Come to Those Who Hustle.: Looking at the often-changed menu, it is quickly apparent that dishes are adventurous and atypical - roasted salmon fish head is offered in proud defiance of squeamish palettes. We don't order it, despite the server's recommendations, but our table of four agrees upon sharing the majority of the dishes available, tapas-style. Hummus arrives served with grilled flatbread and some flowery bitter greens. It is dark and full of spices, cinnamonney and flavorful. It would be great if there was more of the spread to go with all the bread, but who can complain for . Tempura cod tongues are less fishy than expected, meltingly fatty and tasting of the sea, in a perfect light batter and skillfully paired with kimchi to achieve a wonderful umami balance. rwell cove mussels, small and tender, are devoured in their spicy green-juniper jerk sauce, which is first carefully spooned over the mollusks then greedily sopped up with Portuguese corn bread. It is refreshing to experience such a commonly standardized dish with a novel and daring flavour combination.Pieroghis are masterfully executed: the sharp mimolette cheese, sebum edible flowers, goat's butter, pink beets, and young sage leaves create complimentary flavours, simultaneously delicate and fresh yet savory and full of depth. Ricotta gnuddi dumplings, covered in ash and served with nettle masala, are spheres of creamy lightness followed by a subtle, sophisticated complexity of smoke and chlorophyll. nHay-smoked trout is perfectly cooked, the skin crisp and delicious, on a bed of raw baby kale and sorrel-filled fresh pasta, bubbling on arrival with a coating of morel foam. Grilled quail is succulent, perhaps cooked in a process more involved than basic grilling, with slender roast parsnips and burnt roses elevating an otherwise seemingly simple dish.Fried chicken, somewhat disappointingly after such an impressive display of culinary prowess, is just that, and though the drumsticks are delicious and not overly greasy in their golden batter, and the whipped potatoes are creamy and perfect, and the bitter green token salad yet another example of skillful counterflavour, the dish is somehow a flat tone in the evening's gastronomic symphony.The first dish not shared was dessert: each had their own excellent rhubarb tart, with a firm yet-not-too-tough whole wheat crust, the soft tart filling adorned with a light cream and more edible flowery goodness, with the only objection to them being the generous diameter - perhaps we should have shared, as we did to sample a tobacco brulee, which, even with it's perfectly textured filling, did not appeal to most with its strong lingering aftertaste and nicotine-induced buzz. It's probably perfect for smokers.Service is friendly and well-intentioned, but lacking in polish - serving utensils are absent on arrival of most dishes, and must be asked for, or, often, obtained ourselves from beside the bar when unable to catch the server's attention; our timing of asking for new plates and cutlery is a careful calculation involving how residual flavours from previous dishes may affect those coming up; knowledge of ingredients and cooking methods does not stand up well to basic questioning, and promises to follow-up after asking the kitchen are not fulfilled. However, these miscues in service, which actually do little to detract from a wonderful experience, cannot be solely attributed to the single waiter, nor the increasing demand on her time as the night progresses and the restaurant fills to

near-capacity, but are perhaps reflective of the whimsical and creative approach of proprietor and chef Nathan Isberg. The components of the dishes often differ slightly from those portrayed on the menu, giving the sense that the words on paper are just a conceptual framework, with ingredients skillfully substituted and modified as seen fit. At the Atlantic, the food is art, stripped of intentional pretension and high price; Isberg is, without doubt, a talented

Notemos que si bien, el texto parece ser mas corto, puede contener menos palabtas repetidadas.

1.2 Indice de Jaccard

1.2.1 Dessarrollo

Para calcular esto, necesitaremos el total de locales visitados por dos usuarios y necesitaremos los locales que visitaron en común. Para esto, iniciamos con un mapper que contiene el id del usuario y el local que visito. Una vez tenemos esto, el siguiente reducer lo que realizara es sacar un conteo de la cantidad de locales totales que reviso el usuario. Aprovechando de que en este reducer tenemos la lista total de locales vistar por cada usuario, aprovechamos de retornar cada local que reviso un el un usuario acompañado de el id del usuario y la cantidad de locales que este reviso.

```
def mapper_file(self, _, line):
    review_id = line['business_id']
    user_id = line['user_id']
    yield user_id, review_id

def reducer_users_id(self, user_id, movies_ids):
    aux_gen = list(movies_ids)
    total_movies = len(aux_gen)
    for movie_id in aux_gen:
            yield movie_id, [user_id, total_movies]

    # user_id_1 = 4
    # local_id_user_1 = 11
    # local_id_user_1 = 10
    # mapper 1 = 4,11
    # mapper 2 = 4,10
    # total_local_reviews_for_user_1 = 2

    # reducer_1 = 10, [4,2]
    # reducer_2 = 11 , [4,2]
```

Una vez teniendo esto, procederemos a calculas los locales en común que revisaron dos usuarios. Para esto, en el siguiente reducer agrupamos por cada review, la cantidad de personas que realizaron un review. Lo que retornara este reducer sera el id del local y como valor una lista de usuarios que evaluaron este local. Además de el id de los usuarios, tendrá el total de locales que este ha visitado.

```
def reducer(self, movie_id, users_id):
    temp_list = list(users_id)
    yield [movie_id, temp_list]

    # user_id_1 = 4
    # total_local_reviews_for_user_1 = 2
    # user_id_1 = 7
    # total_local_reviews_for_user_1 = 10
    # local_id = 113

    #reducer = 133, [[4,2],[7,10]]
```

Una vez tenemos esto, armamos tuplas de usuarios para posteriormente, poder saber el total de peliculas que vieron en comun. En este caso, el mapper lo que hara sera tomar un par de usuarios, acompañados de el total de peliulas que estos vieron por separado y les asignara un uno como valor, para que así, al momento de reducirlos, podremos saber la cantidad de pelicuals que vieron en comun.

Una vez que el el maper termina, el reductor se encarga de calcular el indice. Para esto, suma el total de peliculas que tienen en comun y las divide por la suma de peliculas totales que revisaron cada uno independiente,ente.

```
def mapper_set_users(self,movie_id, users_ids_extended):
    if len(users_ids_extended) > 1:
            for pair_of_users in itertools.combinations(users_ids_extended, 2):
                    yield  [pair_of_users, 1]

def reducer2(self, key, values):
    user1_total = key[0][1]
    user2_total = key[1][1]
    total = user1_total + user2_total
    yield ['MAX',[float(sum(values))/float(total)]]
```

Para finalizar, el ultimo reductor se encarga de encontrar el maximo indice entre dos usuarios.

```
def max_reducer(self, stat, values):
      yield [stat,max(values)]
```

1.2.2 Resultados

En cuanto a esta pregunta, los resultados no fueron los esperados, debido a que en ambos sets de datos, los resultados al parecer no dan lo que deberían. Por un lado, con 10.000 datos, tenemos el el mayor índice es de 0.5. Por otro lado, con 30.000 el mayor es 1. Estos resultados pueden deberse a la estructura de los datos trabajados, efectivamente puede pasar de que dos personas solo hayan evaluado un local y ese puede ser el único local que tengan en común, por lo que el el el resultado podría no ser del todo raro. En cuanto a los tiempos de ejecución, el set de 10.000 tardeo 51.75 segundos, mientras que el de 30.000 tardo 1 minuto 31 segundos.

1.3 P3

2 P4

2.1 Dessarrollo

Para iniciar, calcularemos la primera parte, que sera el numero total de estrellas otorgados por usuario a cada documento.

$$\sum_{i=1}^{N} A_i$$

Para esto, creamos el siguiente mapper acompañado de su respectivo reducer:

```
def mapper_file(self, _, line):
    review_id = line['business_id']
    user_id = line['user_id']
    star = line['stars']
    yield user_id ,[star, review_id]


def reducer_users_id(self, user_id, movies_ids):
    total = 1
    aux_gen = list(movies_ids)
    for x in aux_gen:
            total = int(x[0])*int(x[0])*total
    for movie_id in aux_gen:
            yield movie_id[1], [user_id, total, movie_id[0]]
```

En esta primera instancia, nos preocuparemos de encontrar el total de estrellas otorgadas por usuario a todos los locales. El mapper nos entrega el id del usuario, acompañado por las estrellas otorgadas y el id del local que evaluó.

```
# user_id = 4
# estrellas otorgadas = 2
# id del local que se evaluo = 44
# retorno mapper = 4, [2,66]
```

Una vez tenemos esto, procedemos a juntar el total de estrellas por cada local. Para esto, se utilizo el reductor, que consiste en transformar en una lista el iterador entregado, para posteriormente recorrerlo y poder saber el numero total de estrellas otorgadas por el usuario en cada local que evaluó. Por ultimo, lo que retornara este método sera el id del local, mas tres valores como iterador.

El primer valor ser el id del usuario, el segundo sera el valor total calculado anteriormente y por ultimo, la cantidad de estrellas que el usuario le otorgo al locala que se utiliza como llave.

Lo segundo que se hizo fue aplicar otro reductor que no ayudara a agrupar a los usuarios por locales que evaluaron.

```
def reducer(self, movie_id, users_id):
  temp_list = list(users_id)
  yield [movie_id, temp_list]
```

El resultado de esto, sera el id del local, acompañado por todos los usuarios que realizaron un review en el. Cabe destacar que no solo trae a los usuario, sino que también la información mencionada anteriormente, esto es: total de estrellas y puntuación individual de dicho local.

```
# local_id = 66
# user_id_1 = 4
# user_id_2 = 7
# estrellas otorgadas por el usuario 1 = 3
# estrellas otorgadas por el usuario 2 = 4
# puntuacion total usuario 1 = 33
# puntuacion total usuario 2 = 10
# retorno reducer = 66, [[4,33, 3], [7, 10, 4]]
```

Luego, se procedio a calcular :

$$\sum_{i=1}^{N} A_i * B_i$$

Para esto se utilizo otro conjunto de mapper y reducer

```
def mapper_set_users(self,movie_id, users_ids_extended):
    if len(users_ids_extended) > 1:
            for pair_of_users in itertools.combinations(users_ids_extended, 2):
                    rev1 = pair_of_users[0][2]
                    rev2 = pair_of_users[1][2]
                    yield pair_of_users, rev1*rev2

def reducer2(self, key, values):
    user1_total = math.sqrt(key[0][1])
    user2_total = math.sqrt(key[1][1])
    lower = user1_total * user2_total
    yield ['max',float(sum(values))/float(lower)]
```

La función del mamapper es elaborar tuplas de usuarios por cada local. Recordar que el resultado de lo anterior es el id del local como key y como valor una lista con la informacion de los usuarios. El resultado de este mapper sera, la tupla de usuario como llave, y la multiplicaccion de las estrellas que cada uno le otorgo al lugar. Recordar que la tupla de los usuarios contiene tanto el id de los usuario como la cantidad total de estrellas otorgadas y la puntuación individual de estos.

```
# retorno reducer = 66, [[4,33, 3], [7, 10, 4], [3,5,2]]
# tupla_1 = [[4,33, 3], [7, 10, 4] ], 3*4
# tupla_2 = [[4,33, 3],[3,5,2]], 3*2
# tupla_3 = [[7, 10, 4],[3,5,2]], 4*2
```

El reductor sera quien junte todo. Por temas de conveniencia, se aplicaran las raizes y otros elementos de la formula mencionada anteioremnte. Para finalizar, se aplica el ultimo reductor, que nos netrega el indice de la tupla que mas similitud tiene

```
def max_reducer(self, stat, values):
    TEMP = [values]
    yield [stat,max(values)]
```

2.2 Resultados

Cando se probo con el set de 10.000 datos, tomo un tiempo aproximado de 32.562 segundo, mientras que cuando se probo con 30.000, tarde 1 minuto 45.362 segundos. En ambos casos se llego al mismo resultado, que el índice mayor es dos. Claramente esto es muy poco usual y se puede deber a algún mal calculo realizado. El posible error podría ser al momento de calcular el total de estrellas realizados por usuario, debido a que esa parte del código es la que arroja resultados mas extraños. Además, el tiempo de computo fue menos al del índice de Jaccard, por lo que podría haber alguna parte del código comprometida con un mal algoritmo.

No obstante, notamos que los resultados son superiores a los del índice de Jaccard. En primer lugar, el índice de Jaccard puede darle importancia a resultados no tan relevantes. Por ejemplo, si dos personas solo asistieron a un local, y ese es el local común, el índice nos entregara una similitud perfecta con este usuario, no obstante, este valor no nos significa nada debido a que la información con cual se calculo el índice era muy poca.