[CS 838 - Spring 2017] Stage 4 Report

Tarun Bansal, Ayush Gupta, Rohit Damkondwar

1. Combination Algorithm

   Schema Matching:
   Both tables A and B have the almost same schema.
   A(Yelp) Schema:
   a. ID
   b. Restaurant Name
   c. Address
   d. City
   e. Zipcode
   f. Latitude
   g. Longitude
   h. Review_count
   i. Rating
   j. Yelp_id

   B(Zomato) Schema:
   a. ID
   b. Restaurant Name
   c. Address
   d. City
   e. Zipcode
   f. Latitude
   g. Longitude
   h. Review_count
   i. Rating
   j. Zomato_id

   Since, schema is almost same (Yelp has extra Yelp_id and Zomato has extra Zomato_id columns), Schema was pretty easy.

   Data Merging :

   ID : Since the ID's in both Table A and Table B is unique and identical, we decided to pick the ID from Table A always. (Token #0)

   Restaurant Name : Select the name with maximum length from left and right tables.

   Address : Select the address with maximum length from left and right tables.

   City : It was required to ensure that the merged city name did not contain the area (suburb) name. Check if any of the tuples already merged had a similar city name. If yes then include that city name in the merged table. If city name exists in the corresponding address, it might not be an area name

Zipcode : Since the zipcode in both the tables is identical. we decided to select the zipcode from Table A always (Token #6)

Latitude : Take the average values of latitude from both the tables
Longitude : Take the average values of longitude from both the tables
Review_count : Add the review counts in both the tables and copy the value to final merged table
Rating : Calculate the aggregate rating using weighted sum of ratings from both sources.
Yelp_id : Copy yelp_id as it is from Table A to the merged table.
Zomato_id : Copy zomato_id as it is from Table B to the merged table.

2. Table E schema
   Table E contains following columns:
   a. Auto generated ID
   b. Restaurant Name
   c. Address
   d. City
   e. Zipcode
   f. Latitude
   g. Longitude
   h. Rating
   i. Source ID

   Total Number of tuples in E: **730**

3. Sample Tuples

1464	Bonfyre American Grille	2601 West Beltline Highway$*$ Madison$*$ WI 53713	Madison	53713	43.03	-89.42	754	4.0	17503464	2YlUn3s132hNq5ueGeIiJg
1937	Presti's Bakery & Caf	12101 Mayfield Rd$*$ Cleveland$*$ OH 44106	Cleveland	44106	41.50	-81.59	779	4.27	16962390	orrrhqRRUORIzUSxWTveKg
11357	131 Main	9886 Rea Road$*$ Charlotte$*$ NC 28277	Charlotte	28277	35.0341674	-80.80	811	4.14648582	17148614	110iMPMPEEjFlf8HKVq84g
11882	Ilios Noche	11508 Providence Road$*$ Suite I$*$ Charlotte$*$ NC 28277	Charlotte	28277	35.05361881	-80.77	890	4.078089888	17147444	c-l4nDPZcEwapEiV-Xf08w

4. Python code

```
import json
import os

with open("output.csv","r") as infile:

        # City map to keep track of the city names in the merged table.
        cityMap = {}
```

```
	counter = 0
	for line in infile:
		if (counter == 0):
			print
"ID,name,address,city,zipcode,latitude,longitude,review_count,rating,zomato_id,yelp_id"
			counter += 1
			continue
		output = ""
		listTokens = line.split(",")

		#Check if the id is equal in left and right tables.
		if (listTokens[0] == listTokens[1]):
			output += listTokens[0] + ","

		#Select the name with maximum length from left and right tables.
		if (len(listTokens[4]) >= len(listTokens[14])):
			output += listTokens[4] + ","
		else:
			output += listTokens[14] + ","

		#Select the address with maximum length from left and right tables.
		if (len(listTokens[11]) >= len(listTokens[21])):
			output += listTokens[11] + ","
		else:
			output += listTokens[21] + ","

		''' Check which city name is better.

			Check if any of the previous tuples had a similar city name.
			If yes then include that city name in the merged table
			If city name exists in the corresponding address,
			it might not be an area name
		'''
		if (listTokens[5] != listTokens[15]):
			#print "Different city found in ",listTokens[5]," and
",listTokens[15],"Addresses : ",listTokens[11]," and ",listTokens[21]
			if listTokens[5] in cityMap:
				if listTokens[15] in cityMap:
					if cityMap[listTokens[5]] >= cityMap[listTokens[15]] :
						output += listTokens[5] + ","
					else:
						output += listTokens[15] + ","
				else:
					output += listTokens[5] + ","
			else:
				if listTokens[15] in cityMap:
					output += listTokens[15] + ","
				else:
```

```
                    if (listTokens[5] in listTokens[11]):
                        output += listTokens[5] + ","
                    elif (listTokens[5] in listTokens[21]):
                        output += listTokens[5] + ","
                    elif (listTokens[15] in listTokens[11]):
                        output += listTokens[15] + ","
                    elif (listTokens[15] in listTokens[21]):
                        output += listTokens[15] + ","
        else:
            if listTokens[5] not in cityMap:
                cityMap.setdefault(listTokens[5],1)
            else:
                cityMap[listTokens[5]] += 1
            output += listTokens[5] + ","

        #Zipcode should be equal as it is the blocked attribute. Hence extract it from
any table
        output += listTokens[6] + ","

        #Take the average values of the latitude and longitude from both the tables
        if ((float(listTokens[8]) != listTokens[18] or listTokens[9] != listTokens[19])):
            output += str((float(listTokens[8]) + float(listTokens[18])) / 2) + ","
            output += str((float(listTokens[9]) + float(listTokens[19])) / 2) + ","

        # Add the review counts in both the tables and copy the value to final table
        total = int(listTokens[7]) + int(listTokens[17])
        output += str(total) + ","

        # Calculate the aggregate rating using weighted sum of ratings from both
sources.
        combined_rating = 0
        if (total != 0):
            combined_rating = (float(listTokens[7])*float(listTokens[10]) +
float(listTokens[17])*float(listTokens[20]))/total
        else:
            combined_rating = int(listTokens[10])
        output += str(combined_rating) + ","

        #Copy zomato_id and yelp_id as it is.
        if(int(listTokens[12]) == 0):
            output += str(listTokens[22]) + ","
        else:
            output += str(listTokens[12]) + ","

        if(listTokens[13] == 0):
            output += str(listTokens[23]) + ","
        else:
            output += str(listTokens[13])
        print output
        counter += 1

#print cityMap
```