Stage 2 : Blocking

Team members:

Anup Rathi
Atasi Panda

Sites crawled:

Yellow Pages
Yelp

Attributes:

Yellow pages:
name, address, city, state, zipcode, phone number, website
Yelp:
name, address, city, state, zipcode, phone number

Number of tuples:

Yellow pages:
11840 rows x 9 columns
Yelp:
5223 rows × 7 columns

Blocking:

First we wanted to block on phone numbers since if phone numbers match, there is a very high probability that the restaurants must be the same real life entities. But due to few missing phone numbers, we were losing matching tuples. Then we used an equivalence blocker on names but that would result in low precision because our data set has a lot of chains so there will be multiple different restaurants with the same name . It would also reduce recall because some restaurants have an added word like restaurant and all such matching tuples were getting dropped. Using overlap on restaurant names returned a huge candidate set because a lot of them have the word, restaurant, which didn't help in getting rid of irrelevant tuples.
So, first we blocked on phone numbers and took the union of the resulting set with the set we got after blocking on zip code. However, blocking on zip codes also returns a lot of irrelevant tuples because there are many restaurants in the same area. So, we took the union after blocking on phone number and on address on the original sets. And on this candidate set we blocked on names using overlap blocker with size 1.

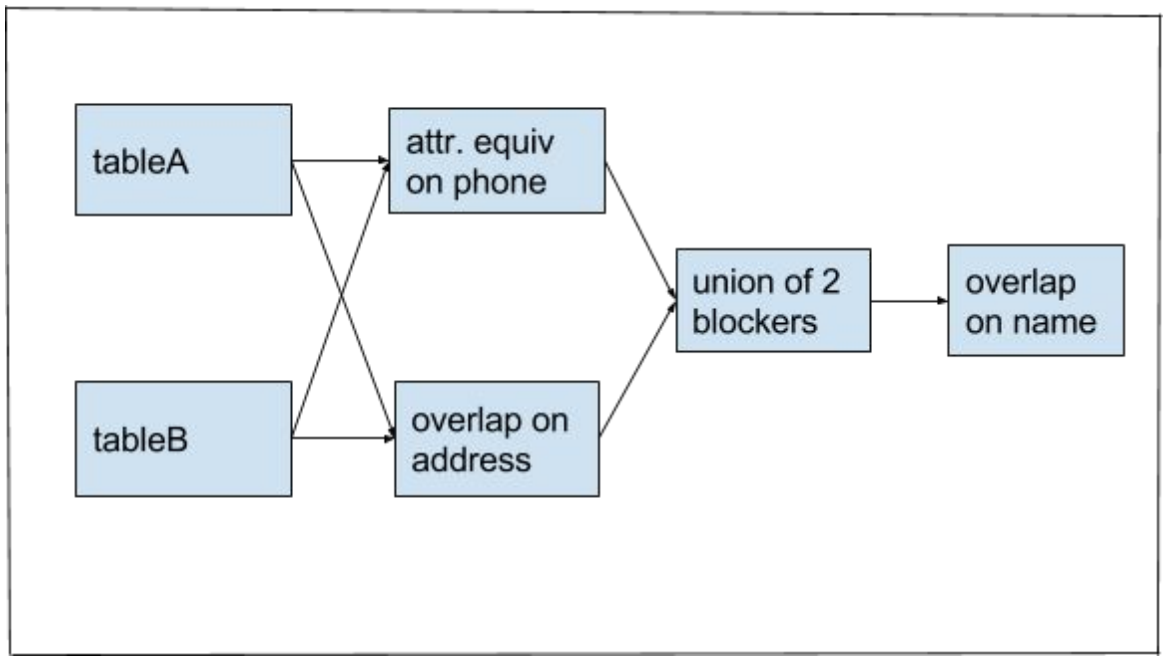
Attempts:

1. Overlap blocking on names with a size of 3 – 1,100,000 tuples
2. Equivalence blocking on phone numbers – 4,000 tuples
3. Equivalence blocking on names - 4503 tuples
4. Union of equivalence blocking on phone numbers and equivalence blocking on zip codes – 300,000 tuples

Final Blocking:

1. Equivalence blocking on phone numbers
2. Overlap blocking on address with size = 3
3. Union of the resulting sets from step 1 and 2
4. Overlap blocking of size 1 on names on the candidate set that resulted from step 3

Blocking diagram:



Debugger -

We used debugger to ensure we are not dropping any potential tuple pair that matches. Using debugger also helped us to place different blockers in order. For example earlier we used overlap blocker on name on table A and B but debugger showed that output still

contains lot of redundant matches. Hence we applied overlap blocker on name in final stages.

It took us about 2 days to finish the blocking step. We spend much of the time in understanding how the API's work. Sample program on Pradap's page finally helped us.

Sizes –

1. Table A - 11840 rows
2. Table B - 5223 rows
3. AXB - 61840320 tuples
4. Final set after blocking - 5278 tuples

Additional Cleaning and Extraction -

We had blank lines in the csv file after extracting data from yelp, which were just extra rows with no useful data. This could have created lots of false matching pairs if the blocker allowed missing value tuples to be retained. We got rid of it using methods in microsoft excel.

Feedback -

1) As discussed in class, having an option to keep the missing values as a match would be great.
2) We faced some problems with figuring out few syntaxes. The documentation could be improved. For example, AttrEquivalenceBlocker() is mentioned as AttrEquivBlocker() in the document. Although this problem was easily overcome by using tab on ipython notebook. But having an example for the commonly used methods along with parameters would be really helpful.