DEPARTMENT OF COMPUTER SCIENCES, UNIVERSITY OF WISCONSIN-MADISON

Report for CS 784 Project Phase 2

Shruthi Racha, Ashish Shenoy
November 1, 2015

1 BLOCKING PROCESS

The main purpose of this process is to bring down the number of tuples to be matched in the two tables, tableA and tableB.
tableA, which was obtained by crawling Zomato, had 3014 tuples. tableB, which was obtained by crawling Yelp, had 5883 tuples. The cartesian product of these two tables had 17731362 tuples which is too large a data set to apply an entity matching algorithm.
We performed the steps as described below to bring this number down.
The entire blocking process took around 5 hours.

1.1 STEP 1 - CLEANING THE DATA

Several tuples in the two tables had missing PHONE_NUMBER entries and these were recorded as "Not Available". This led to a match of all the tuples that had "Not Available" in the PHONE_NUMBER column of tableA with those with the same value in tableB.
Here we could have opted one of the following strategies :

- Include the matches of the "Not Available" phone numbers in our candidate set.
- Block the entries with "Not Available" phone numbers in our candidate set. But perform blocking on a different attribute and combine the results from both the candidate sets.

Since the number of "Not Available" phone numbers were too many in both the tables (Over 25 in tableA and over 50 in tableB) and to avoid having too many false positive matches we decided to assign a unique number to the unavailable phone numbers in both the tables and block those entries by performing an Attribute Equivalence Blocking.
The python code we used to do this is as shown below :

```
phonenumber = soupify.find("span", { "class" : "biz-phone"})
if(phonenumber is not None):
    phonenumber = phonenumber.findAll(text=True)[0].strip()
else :
    phonenumber= unicode(current_milli_time()) \\this ensures a unique number is
                                               \\assigned to empty phone numbers
```

1.2 Step 2 - Attribute Equivalence Blocker on PHONE_NUMBER attribute of tableA and tableB

After we had a cleaned up data set, we decided to bring down the number of tuples to be matched by first applying one of the two inbuilt blockers in Magellan. We used the Attribute Equivalence blocker on the PHONE_NUMBER attribute present in the two tables.
Since the inbuilt blockers in Magellan are optimized to run fast, we thought this would be a good strategy to use as the first stage of blocking.
We chose the PHONE_NUMBER as the attribute for blocking because if two tuples in the tables have the same phone numbers then we are certain that they represent the same real world entity, which here is a restaurant.
We observed from the resulting candidate set that this strategy also took care of scenarios where the restaurant name has changed and is not up-to-date in one of the websites but the phone number has remained the same.
On the other hand we also observed that some of the phone numbers in tableA were not up-to-date with the latest phone number of that restaurant. Hence these tuples were getting blocked even though they were the same real world entity. To tackle this we decided to apply blocking on other attributes, such as Address, and combine the candidate sets to avoid missing out on matches.

1.3 Step 3 - Overlap Blocker on ADDRESS attribute of tableA and tableB

The second blocker we used was the Overlap Blocker on the ADDRESS attribute of tableA and tableB. We started with an overlap size of 2 at word level and observed from the resulting candidate set that this was too lenient a measure.
After checking a sample of tuples, we decided that an address can be considered to be a match if atleast 4 words overlap between them.
Since the addresses will almost always overlap on the city and state (Eg. Madison, WI are common to all restaurants in Madison in both the tables), an overlap of 4 is a reasonable, not too strict and not too lenient constraint for tuples to blocked.
This stage brought down the number of tuples to be matched from 17731362 to 88703 tuple pairs and the candidate set was named : *Address_Overlap_CandSet*
An example entry in the two tables with their addresses is as shown below :

```
Zuzu Cafe  3.9 (608) 260-9898 60 1336 Drake St, Madison, WI
ZuZu Cafe & Market  3.5 (608) 260-9898 39 1336 Drake St, Madison, WI 53715
```

1.4 Step 4 - Rule Based Blocker on the Address_Overlap_CandSet

To further narrow down the number of potential matches in the candidate set obtained after applying the Overlap Blocker, we decided to apply Jaccard on the ADDRESS attribute of the candidate set. For this we used the Rule Based Blocker in Magellan with the following rule :

```
ADDRESS_ADDRESS_jac_qgm_3_qgm_3(ltuple,rtuple) < 0.44
```

To arrive at the value of 0.44, we used the Debug Blocker in Magellan to debug, understand and verify the consistency of the similarity scores for a sample of the tuple pairs with the values obtained by manual calculation. The outputs from the debug blocker helped us get a sense of what was happening in the rule based blocker.
After this stage we obtained a candidate set : *Address_Overlap_Jac_CandSet* with 25303 tuple pairs.

1.5 Step 5 - Overlap Blocker on NAME Attribute of tableA and tableB

In all the above steps we performed blocking either on ADDRESS attribute or the PHONE_NUMBER attribute of the two tables. To leverage the information in the NAME attribute of the tables we decided to also combine the outputs obtained by applying blocking on this attribute.
We decided to write a Black Box Blocker with a custom function to block on the NAME attribute. But to reduce the number of tuple pairs to be sent to Black Box Blocker we decided to first use a word level Overlap Blocker with a overlap size of 1 on NAME attribute of tableA and tableB.
The resulting candidate set from this stage : *Name_Overlap_CandSet*

1.6 Step 6 - Black Box Blocker on Name_Overlap_CandSet

To compare two names in a tuple pair we decided to write a custom function which is a combination of Levenshtein Distance and word level comparision.
The algorithm we used was :

- 1. We arrived at a blacklist of the following words :

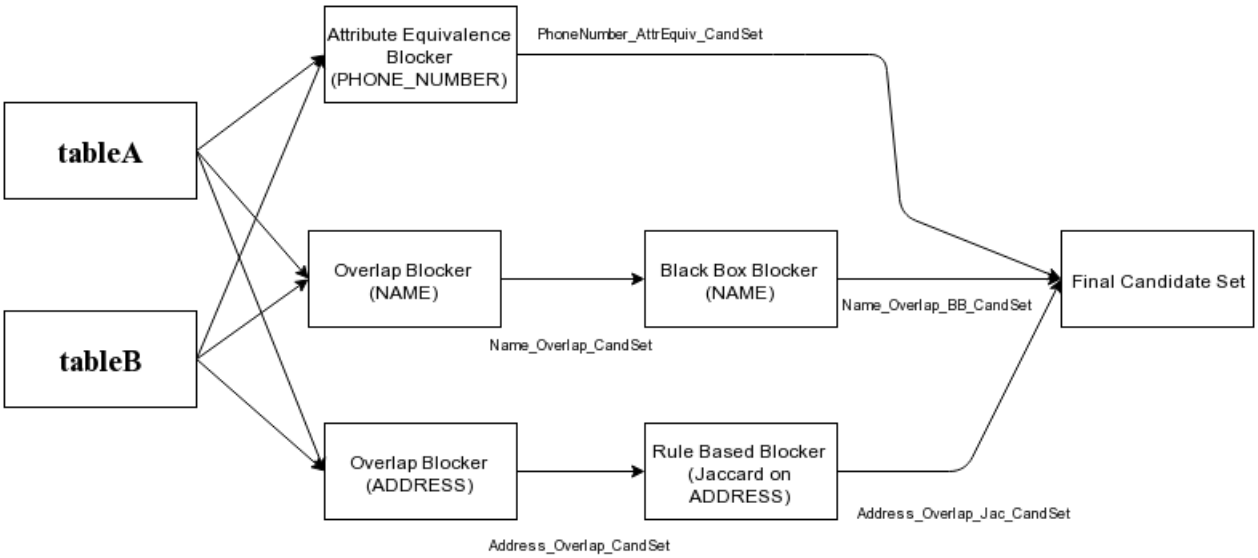
  ```
  black_list = ["for", "the", "of", "a", "an", "and", "&", "on", "cafe",
  "restaurant", "grill", "pizza", "pizzeria", "pub", "bar"]
  ```

- 2. Skip the below steps for the word is in the blacklist
- 3. If atleast one of the words in the name matches return False (i.e., Don't Block)
- 4. Else Calculate the Levenshtein Distance of the two words.
- 5. Return False if Levenshtein distance is less than 3. Else return True.

The resulting candidate set : *Name_Overlap_BB_CandSet* had 53438 tuple pairs.

1.7 Step 7 - Combining the Candidate Sets

The final candidate set was obtained by combining *Name_Overlap_BB_CandSet, PhoneNumber_AttrEquiv_CandSet* and *Address_Overlap_Jac_CandSet* which had a length of 78190.



2 Magellan Feedback

- Magellan could not handle spaces in the attribute names. For example, when we first tried to run the attribute equivalence blocker on our tables, the Phone Number column had space in it and the blocker failed without giving any indication that it failed due to the space in the column name.
- Magellan should be able to handle trailing and leading spaces in the names of the attributes. Considerable time was spent in figuring out why "Name" and " Name" were being recognized as same.
- Better exception handling is necessary to help user understand why something failed.
- The debug blocker should use the same internal algorithm as the one used in the blockers.
- It would be a good to have feature if we could run the Rule Based Blocker and the Black Box Blocker directly on the tables we started out with instead of the candidate sets. Right now these blockers don't scale for such large tuple pairs.
- The documentation for the library should be updated. Some of the function names used in the manual are not consistent with the ones in the code of the library.