CS 839 Spring 2018, Project Stage 3

Team members:
Arpith Neelavara (neelavara@wisc.edu)
Bhargav Tangirala (btangirala@wisc.edu)
Aribhit Mishra (amishra28@wisc.edu)

1. Describe the type of entity you want to match, briefly describe the two tables (e.g., where did you obtain these tables), list the number of tuples per table.
The entity we performed our match was on a set of restaurants in New York City. The data for the tables was taken from two different web sources namely TripAdvisor and Yelp.

2. Describe the blocker that you use and list the number of tuple pairs in the candidate set obtained after the blocking step.
We have used the combination of two blockers:
One blocks based on Name of the restaurant (Jaccard Measure with 3 grams with a constraint 0.3) and Street (Jaccard Measure with 3 grams with a constraint 0.3). We have created a street attribute from address attribute because it has the most sensitive information compared to complete address.
Next blocker is only on the Street (Jaccard Measure with 3 grams with a constraint 0.6).

We combined the results of two different blockers using union for the following reasons.

Street (from Address) only because it can capture some pairs where names are same but differ by a new word. Ex. (alfa ristorante, alfa). The constraint is higher - 0.6

Name only to capture restaurants that have similar names (constraint is 0.3). But added Street based rule on top of that to eliminate chain restaurants with multiple branches at different locations (eg: Shake shacks at Manhattan, Shake Shacks at Brooklyn). The constraint is lower threshold in this case compared to the earlier blocker.

The number of tuple pairs in the candidate set obtained after the blocking step was 1625.

3. List the number of tuple pairs in the sample G that you have labeled.
We have sampled 600 tuples and labelled them.

4. For each of the six learning methods provided in Magellan (Decision Tree, Random Forest, SVM, Naive Bayes, Logistic Regression, Linear Regression), report the precision, recall, and F-1 that you obtain when you perform cross validation for the first time for these methods on I.

Below are the results we have obtained after cross validation:

	Matcher	Average precision	Average recall	Average f1
0	DecisionTree	0.966252	0.963852	0.964790
1	RF	0.980505	0.973049	0.976478
2	SVM	0.817495	0.986258	0.892202
3	LinReg	0.983994	0.943466	0.963085
4	LogReg	0.966219	0.964915	0.965403
5	NaiveBayes	0.974935	0.964915	0.969759

5. Report which learning based matcher you selected after that cross validation.

From the above scores we see that Random Forest gives the best F1 score, we have selected Random Forest.

6.Report all debugging iterations and cross validation iterations that you performed. For each debugging iteration, report (a) what is the matcher that you are trying to debug, and its precision/recall/F-1, (b) what kind of problems you found, and what you did to fix them, (c) the final precision/recall/F-1 that you reached. For each cross validation iteration, report (a) what matchers were you trying to evaluate using the cross validation, and (b) precision/recall/F-1 of those.

Since we obtained good results initially, we didn't have to go through the debugging phase and hence the results are the same as before.

7.Report the final best matcher that you selected, and its precision/recall/F-1.

Same as above.

8.For each of the six learning methods, train the matcher based on that method on I, then report its precision/recall/F-1 on J.

Random Forest:

Precision : 97.37% (148/152)

Recall : 97.37% (148/152)

F1 : 97.37%

False positives : 4 (out of 152 positive predictions)

False negatives : 4 (out of 67 negative predictions)

Decision Tree:
Precision : 93.59% (146/156)
Recall : 96.05% (146/152)
F1 : 94.81%
False positives : 10 (out of 156 positive predictions)
False negatives : 6 (out of 63 negative predictions)

SVM:
Precision : 83.62% (148/177)
Recall : 97.37% (148/152)
F1 : 89.97%
False positives : 29 (out of 177 positive predictions)
False negatives : 4 (out of 42 negative predictions)

Logistic Regression:
Precision : 95.39% (145/152)
Recall : 95.39% (145/152)
F1 : 95.39%
False positives : 7 (out of 152 positive predictions)
False negatives : 7 (out of 67 negative predictions)

Linear Regression:
Precision : 95.51% (149/156)
Recall : 98.03% (149/152)
F1 : 96.75%
False positives : 7 (out of 156 positive predictions)
False negatives : 3 (out of 63 negative predictions)

Naive Bayes:
Precision : 96.73% (148/153)
Recall : 97.37% (148/152)
F1 : 97.05%
False positives : 5 (out of 153 positive predictions)
False negatives : 4 (out of 66 negative predictions)

9. Report approximate time estimates:
(a) to do the blocking - 10 seconds (to run the blocker)
(b) to label the data - 1.5 hour
(c) to find the best matcher - 76 seconds (running the select matcher and getting the results)

10. Provide a discussion on why you didn't reach higher recall, and what you can do in the future to obtain higher recall.

We have reached a recall of 95+ which seems to be a very good score.

11.BONUS POINTS: provide comments on what is good with Magellan and what is bad, that is, as users, what else would you like to see in Magellan. Are there any features/capabilities that you would really like to see being added? Any bugs?

Inside rf.predict() function, data frame of generated feature table was not getting accepted as input where we had to give the table and we got catalog error so, we had to convert feature table into CSV and then give it as an input. It would be good if we can directly give the data frame as the input rather than converting it into CSV and then giving it as an input.