PROJECT STAGE 3 – ENTITY MATCHING

Team 29

The entity we are trying to match is books. We extracted these tables from two web sources, goodreads.com and amazon.com, based on rule-based wrapper construction. The attributes of the two tables are:

- title - (name of the book)
- author
- rating
- format - (eg., paperback, hardcover, etc.)

Each table contains 3000 tuples.

We used the overlap and black box blocker. Overlap blocker with overlap_size=3 and attribute *'title'* generated 54,723 tuple pairs. We reduced this number by using block_candset. Here, we assumed that two books with no overlap of their authors cannot refer to the same book. So, for block_candset we chose the overlap_size=1 and attribute *'author'*. This reduced the tuple pairs to 548. Debugging the blocker we found that few matching tuples pairs could not survive blocking. Hence, we used a black box blocker to generate those tuple pairs missed by overlap blocker and combined the results (i.e., union) of these two blockers. Thus, the total number of tuple pairs which survived blocking is 565.

The number of tuple pairs in the sample G that you have labelled – 565

Cross validation for the first time for the learning methods on set I:

MATCHER	**AVERAGE PRECISION**	**AVERAGE RECALL**	**AVERAGE F1**
Decision Tree	0.885933	0.880029	0.881771
Random Forest	0.901088	0.903844	0.901685
SVM	0.915229	0.839137	0.875228
Linear Regression	0.883766	0.929070	0.904290
Logistic Regression	0.856335	0.918478	0.884201
Naïve Bayes	0.802324	0.877938	0.837078

We selected Random Forest after CV because it has the highest F1 score.

As we had achieved the desired precision and recall, no debugging and cross validation iterations were performed.

The final best matcher which we selected was Random Forest.

- Precision: 0.901088
- Recall: 0.903844
- F1: 0.901685

Precision/Recall/F-1 on J:

MATCHER	AVERAGE PRECISION	AVERAGE RECALL	AVERAGE F1
Decision Tree	100% (183/183)	100% (183/183)	100%
Random Forest	98.92% (183/185)	100.0% (183/183)	99.46%
SVM	94.54% (173/183)	94.54% (173/183)	94.54%
Linear Regression	88.21% (172/195)	93.99% (172/183)	91.01%
Logistic Regression	86.29% (170/197)	92.9% (170/183)	89.47%
Naïve Bayes	80.1% (161/201)	87.98% (161/183)	83.85%

Final matcher Y is Random Forest. The scores on J are:

- Precision : 98.92%
- Recall : 100%
- F1 : 99.46%

Approximate time estimates:

a) For blocking – Overlap blocker gave results instantly whereas the black box blocker ran for approximately 5 minutes and 41 seconds.
b) Labelling the data – As we had 565 tuple pairs after blocking, labelling them took around 45 minutes.
c) Finding best matcher – As we were able to achieve the desired precision/recall/f1 scores in the first iteration of cross validations, finding the best matcher did not take any time.

Q:) Discussion on why we didn't reach higher recall, and what you can do in the future to obtain higher recall?

A:) The recall of our best matcher (Random Forest) on set I was around 90.3% and set J was 100%. Hence, we did not improve our recall further.

<u>Comments on Magellan:</u>

Highlights (the good part):

1. Easy to use, clean pipeline for entity matching.
2. Provides great accuracy within less time.
3. Semi-automation of feature generation thereby eliminating the biggest task of the user.
4. Is Magellan deployed for commercial usage? If yes, how is the framework deployed as Python is not meant for production use-cases?

Highlights (the buggy part):

1. Bugs – py_entitymatching got installed on Anaconda using the command *'conda install -c uwmagellan py_entitymatching'.* But after installation the py_entitymatching package was empty in anaconda's installed packages. (Windows)

Highlights (the possible improvements):

1. **Sampling**: The sampling method is expensive currently. Can we use stratified sampler or sampling by clusters?
2. Can we learn **blockers** by discovering correlations and functional dependencies?
3. **Blocker optimization**: Currently, the blocking usage reiterates the overlapping of tuples for every rule. This is redundant. Can we optimize this process by saving the overlapping results and introduce parallelism for blocking rules which are independent by using daskframe or similar techniques?
4. Has Magellan been tested for following use-cases:
    a. Attributes with text. How is semantic similarity taken care here?
5. Can Magellan extend to map dissimilar schema?
6. As a user, I would like to eliminate labelling task in Magellan for future use. (Unsupervised version of Magellan)