CS 838 — Data Science: Principles, Algorithms, and Applications; Spring 2017

Stage 3: entity matching

Trang Ho, Thomas Ngo, Qinyuan Sun

1. Introduction

In this project stage, our team performed matching entities between two tables of education affiliations. The first table was extracted from the Academy of Management Conference (AOM) website, which contains personal information of the conference attendants in the year 2014. The personal information includes (1) individual name, (2) affiliation name, (3) country, (4) states/ province, (5) city, (6) contact numbers, (7) email address. Overall, this table consists of 9,532 entities at the individual level.

The second table was extracted from the World Higher Education Database (WHED), which contains information of unique education affiliations worldwide. This table provides information on (1) affiliation name, (2) country, (3) street, (4) city, (5) province/ states, (6) postal code, (7) telephone number, and (8) website address (if available). Overall, this table consists of 17,605 unique entities at the affiliation level.

In order to match individuals' affiliations on the first table to affiliations on the second table, we used their overlapped/relevant information: (1) affiliation name, (2) country, (3) province/states, (4) city, (5) website address, (6) individual email address. Our goal here is to get precision score of above 95% and recall score of as high as possible.

Subsequently, we carried out the following steps using Magellan:

- Pre-processing
- Down-sizing the AOM table and the WHED table
- Using a blocker to reduce the size of the potential-candidate set
- Sampling randomly 500 pairs of potential candidates for labelling
- Creating training and testing sets I and J
- Training and selecting the best classifier using cross-validation
- Evaluating performance on the testing set J

More details can be found below.

2. Matching procedure

Step 1. Pre-processing

In this step, we cleaned the two datasets by standardizing information on affiliation names, country, state/province, city, email server domain. For example, we standardized states by transforming "CA", "CA - California", "California" to "california" on both the AOM table and the WHED data.

Step 2. Down-sizing

Initially, we have 9,532 entities on the AOM table and 17,605 entities on the WHED data. After down-sizing, we have 4,000 AOM entities and 4,962 WHED entities

Step 3. Blocking

Our blocking consists of the following components:

- Blocking all tuple pairs that have different countries
- For American affiliations, blocking all tuple pairs that have different province/ states
- For all affiliations, blocking all tuple pairs that have neither (1) any overlap between AOM email domain and WHED affiliation website domain nor (2) sufficient overlap coefficient (i.e. greater than 0.5) between affiliation names

As a result, we reduced the size of our candidate set from 19,848,000 (=4,000 x 4,962) to 126,516.

This step took approximately 9 minutes.

Step 4. Sampling for labelling

We initially sampled randomly 500 tuple pairs from the set of 126,516 potential candidates. After labeling, we dropped 22 cases due to ambiguity of the AOM information. Consequently, we had 478 tuple pairs with a density of approximately 34%.

We spent about 5 hours on labelling data.

Step 5. Creating training & testing sets

We split the sample set into training and testing sets. As a result, each set has 239 tuple pairs.

	Num. of entities	**Num. of positive examples**	**Num. of negative examples**
Training Set I	239	81 (33.8%)	158 (66.2%)
Testing Set J	239	73 (30.5%)	166 (69.5%)
Total	478	154	324

Step 6. Training and selecting the best classifiers

We used 6 learning methods for training on set I using 5-fold cross validation. The methods include: (1) Decision Tree, (2) Random Forest, (3) SVM, (4) Naive Bayes, (5) Logistic Regression, and (6) Linear Regression. Our classifiers use 34 features, and below is the first-attempt accuracy performance of our classifiers on the training set I:

Machine Learning Algorithm	**Avg. CV Precision**	**Avg. CV Recall**	**F1**
Decsion Tree	0.88	0.87	0.87
Random Forest	0.95	0.90	0.92
Support Vector Machine	0.96	0.51	0.63
Naive Bayes	0.96	0.51	0.63

Machine Learning Algorithm	Avg. CV Precision	Avg. CV Recall	F1
Logistic Regression	0.89	0.89	0.88
Linear Regression	0.93	0.84	0.88

For its simplicity and accuracy performance, we selected the Random Forest learning method for our testing.

This step took about 1 minute.

Step 7. Evaluating performance

We trained the classifier with all the training examples and tested on the testing examples. The results are shown in the following table.

Type	Precision	Recall	F1
TRAIN	1.00	0.98	0.99
TEST	0.99	0.91	0.95

3. Links

Link (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/src) to source code directory:

- Master source code: (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/src/Stage_3.ipynb) Stage_3.ipynb

Link (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files) to data directory:

- Source table: _aom.csv (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files/_aom.csv), _whed.csv (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files/_whed.csv)
- Preprocessed table: _aom_cleaned.csv (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files/_aom_cleaned.csv), _whed_cleaned.csv (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files/_whed_cleaned.csv)
- Pairs that survive the blocking step: matching_pairs_table_overlap3_emailserver.pkl (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files/matching_pairs_table_overlap3_emailserver.pkl)
- Golden data: golden_data_labeled_nomissing.csv (https://github.com/TrangHo/cs838-spring2017/tree/master/stage3/csv_files/golden_data_labeled_nomissing.csv)

4. Comments on Magellan

Overall, Magellan provides many helpful tools for entity matching. We very appreciate that, although there is a minor feedback that we would like to share. In particular, we had trouble with setting up PyQt4. It seems to be deprecated because the official page of Qt only provides Qt5, and PyPi only provides install formula for PyQt5. After spending a significant amount of time but unsucessfully installed PyQt4, we have forked the git repo for py_entitymatching (https://github.com/anhaidgroup/py_entitymatching) to update the code. As of now, the modified code works perfectly with PyQt5. The forked repo here

(https://github.com/TrangHo/py_entitymatching) and the changes can be found in branch support_pyqt5 (https://github.com/TrangHo/py_entitymatching/tree/support_pyqt5). We will update the documents and make a pull request in the future.

Finally, thank you very much for the useful package!