Stage 3- Entity Matching

Sukanya Venkataraman (venkatarama5@wisc.edu)
Yogesh Chockalingam (ychockalinga@wisc.edu)
Shantanu Singhal (singhal5@wisc.edu)

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 chose was books, and we extracted books from Amazon.com and Goodreads.com. The schema of the tables containing books from these sources is as follows:

S No.	Attribute	Type	Description
1	Name	String	Name of the book
2	Author	String	Name of the author(s) of the book
3	Publisher	String	Name of the book publisher
4	Publishing_Date	String	Date published. Format - year-month-day
5	Format	String	Format of the book - Paperback, Kindle etc.
6	Pages	Int	Number of pages
7	Rating	Float	Book rating, out of 5

The 2 tables are called source1_cleaned.csv and source2_cleaned.csv. They contain 3387 and 3001 tuples respectively.

Describe the blocker that you use and list the number of tuple pairs in the candidate set obtained after the blocking step.

We used a blocking sequence and each step is outlined below:

Rule Based Blocker

We first get the tokenizers and the similarity functions using APIs from *py_entitymatching* (Refer notebook) and then get the attribute correspondence for the two tables.

We then define the following rules:
1. For a tuple pair, if the Levenshtein similarity for the Name attribute is less than 0.275, block them.
2. For a tuple pair, if the Jaccard similarity for the Author attribute is less than 0.5, block them.

Overlap Blocker

We then applied the overlap blocker to the candidate set obtained in the previous step. Since the entity we are dealing with is books, there are quite a few stopwords present in the book names, such as "The", "Of", "And" etc. Hence, we removed these stopwords (Refer Notebook) and then performed overlap blocking with the overlap size set to 1.

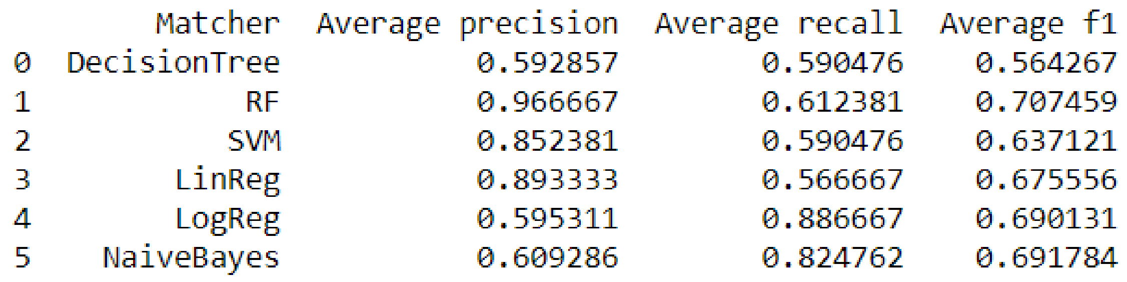
We apply overlap blocking to the following attributes:
1. Book Names
2. Book Authors

Number of tuple pairs obtained after blocking: 1092

List the number of tuple pairs in the sample G that you have labeled.

We labeled 500 tuple pairs to obtain G.

For each of the six learning methods provided in Magellan (Decision Tree, Random Forest, SVM, Naïve 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.



```
        Matcher  Average precision  Average recall  Average f1
0  DecisionTree           0.592857        0.590476    0.564267
1            RF           0.966667        0.612381    0.707459
2           SVM           0.852381        0.590476    0.637121
3        LinReg           0.893333        0.566667    0.675556
4        LogReg           0.595311        0.886667    0.690131
5    NaiveBayes           0.609286        0.824762    0.691784
```

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

We selected the random forest classifier after we performed cross validation, as it had the highest precision and F1 score.

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.

We had an initial precision of ~70% and recall of ~40%, (F1 score of ~0.5) for Random Forest. We decided to debug this matcher for the following reasons –

1. It was still the best performaing matcher
2. The GUI for random forest was available and easy to use.
3. Random forest is one of the easier matchers to debug

Initially, we used only one feature per attribute (except the ID attribute features, which we dropped completely), from the list of automatically generated features. We proceeded with first tackling the problem of precision, for which we looked at the false positives. One of the biggest issues we discovered with our data set was that there were multiple instances of the same book, with different volumes, all being matched to each other after the blocking stage. For example, the book – "The Year's Best Science Fiction & Fantasy, 2009 Edition (Year's Best Science Fiction and Fantasy)" was matched with "The Year's Best Science Fiction & Fantasy 2015 Edition" and "The Year's Best Science Fiction & Fantasy 2016". We tackled this problem by using all the features generated for the 'Name' and 'Author' columns, and dropping the features related to rating. This helped us increase the precision to ~90%, and the recall increased to ~45% as well (Cross validation precision and recall), with an F1 score of 0.6.

We then decided to tackle the problem of recall. We looked at the false negatives and realized two things – There were some cases where the label itself was wrong, and there were some cases where it was falsely being recognized as negative even though the names were an exact match (but sometimes the publishers were different). For this, we tried doing two things –

1. Created a trigger rule to match all names whose levenshtein distance is 0
2. Removed publisher related features from the feature table

Once we removed the publisher related features from the feature table, we were able to obtain the required (given above) precision, and good recall, and applying the trigger rule after this made no difference to the output.

We also increased the number of iterators to 50 (from 10).

We then re did the cross validation, and achieved the desired precision. The final precision and recall we reached is given above in the table.

The notebook which has the pipeline for debugging can be found here.

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

We selected the random forest classifier.

Precision	96.66%
Recall	61.23%
F1	70.74%

Report the numbers 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.

Classifier	Precision	Recall	F1
Decision Tree	100%	100%	100%
Random Forest	100%	100%	100%
Logistic Regression	57.89%	91.67%	70.97%
Linear Regression	90.00%	75%	81.82%
SVM	88.89%	66.67%	76.19%
Naïve Bayes	50.00%	75.00%	60.00%

Report the numbers for the final best matcher Y selected, train it on I, then report its precision/recall/F-1 on J.

Classifier	Precision	Recall	F1
Random Forest	100%	100%	100%

Report approximate time estimates: (a) to do the blocking, (b) to label the data, (c) to find the best matcher.

(a) The rule-based blocking took 6.324 seconds and overlap blocking took 0.18 seconds. Total time: 6.5 seconds
(b) We labelled 500 tuple pairs. This took roughly 40 minutes.
(c) Since we did not have to debug, the process of finding the best matcher took 9.62 seconds.

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

For the test set, we obtained perfect recall. This doesn’t say enough about the matcher being perfect because –

1. The number of positive examples are small
2. The training set precision and recall weren’t perfect

For the training set recall, we believe that we couldn’t increase it more because of difficult false negatives. For example, there were some book pairs which were the same, but in the ltable, their name was truncated, but the rtable, the name for more descriptive. For example –

“Battle Mage” vs “Battle Mage (An Epic Fantasy Adventure)” or

“Blackmark: An Epic Fantasy Adventure Sword and Highland Magic (The Kingsmen Chronicles) (Volume 1)” vs

“Blackmark (The Kingsmen Chronicles #1): An Epic Fantasy Adventure Sword and Highland Magic”

These cases were difficult to handle, and we couldn’t come up with trigger rules or extra features that handled them.