CS838 Data Science
Course Project Stage Three

Group 7
Hongyi Wang, Hao Fu, Miao Yang

April 1, 2017

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

In this stage, we expect to implement entity matching between two data tables in CSV format provided by Silicon Valley research lab. The first data table is called "movies.csv", which describes properties of different movies (e.g. title and year). Also, the "movies" data table contains more than 1000,000 tuples. The other data table is "tracks.csv", which contains 734,485 tuples, describes tracks with attributes like id (key), title, year, artist name, and etc. We suppose to match the entity in the above two tables that describe the same movie based on attributes of "title (i.e. name of movies)" and "year".

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

At the beginning, we only considered to use "title" (i.e. name of move) and "year" as attributes to implement the entity matching. In specific, we implemented an **AttrEquivalenceBlocker** on attribute "year" followed by a **OverlapBlocker** on attribute "title". However, under this case, although we got good results during the debugging stage, we found that in the sampling stage, the actual matching entities were far less than mismatching entities. Hence we looked into **RuleBasedBlocker** and measured the feature values generated from the **get_features_for_blocking**. Finally, we add one RuleBasedBolceker node into our blockers, like in these lines:

```
rb = em.RuleBasedBlocker()
block_f = em.get_features_for_blocking(sample_movies, sample_tracks)
        rb.add_rule(['title_title_cos_dlm_dc0_dlm_dc0(ltuple, rtuple) < 0.6',
        'title_title_jac_qgm_3_qgm_3(ltuple, rtuple) < 0.6'] ,block_f)
```

The AttrEquivalenceBlocker step was remained the same, followed by parallelized overlap blocking and rule-based blocking. After that, we **Union** the results from both the overlap blocking and rule-based blocking. And, the blocking step we implemented in this project was given in Figure 1. Finally, implementing the blocking, we get a sample G, which contains around 2000 tuples.

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

After getting the sample G, we labeled 400 out of 2000 tuple pairs from it manually.

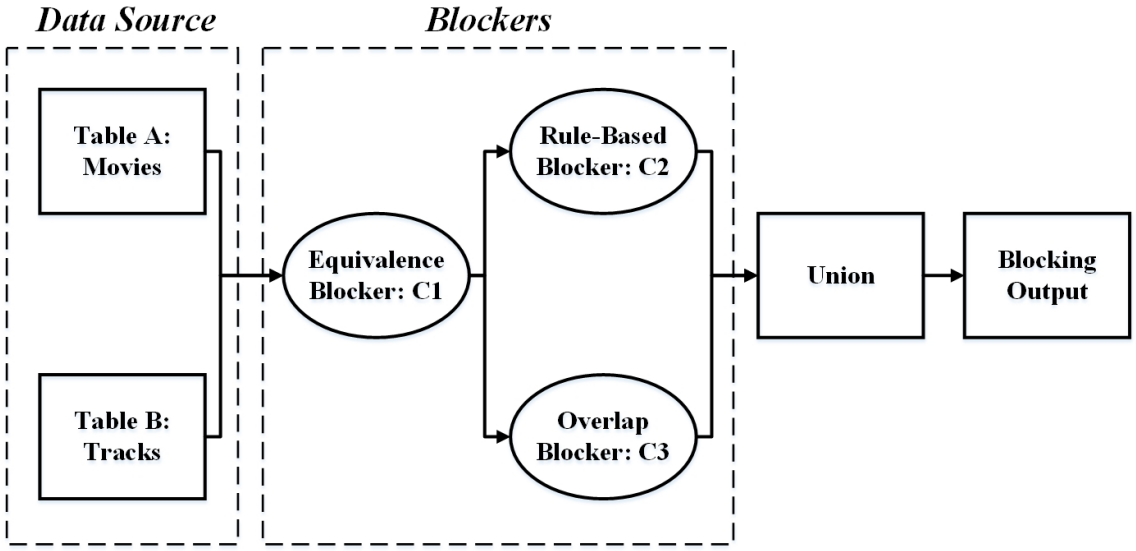

Figure 1: Blocking Steps

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

From the sample G, we labeled 400 tuples, which contains 188 positive labels and 212 negative tuples. We split it into set I and J. We did training and cross validation using set I, and plan to do test on J.

Evaluations Among Ml Models			
ML Models	Precision	Recall	F_1 Score
Decision Tree	0.9049	0.8906	0.8977
SVM	0.9343	0.8957	0.9146
Random Forest	0.8839	0.8455	0.8643
Logistic Regression	0.8546	0.8369	0.8457
Linear Regression	0.8647	0.8307	0.8474
Naive Bayes	0.8347	0.8707	0.8523

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

After implementing cross validation, we found that the SVM is the best classifier given its "Precision", "Recall", and "F-1" were higher than other classifiers.

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.

a) At the beginning, we took sub-attributes (those only involve "time" and "titles") to perform cross validations on the 6 ML models supported in Magellan. At this stage, we found that the SVM classifier get better performance than others, which gain higher than 90% precision.
b) We found that, in the "title" attribute, the string format is really messy, which contains lots of

irrelevant contents. Thus, we did data cleaning on the "title" attributed of "movies" and "tracks" and add rule-based blocker to gain higher performance. After this, we found that the "random forest" classifier gain best performance.
c) Final Results

Evaluations Among Ml Models			
ML Models	Precision	Recall	F_1 Score
Decision Tree	0.9714	0.8914	0.9269
SVM	0.9667	0.8184	0.8844
Random Forest	0.9814	0.9763	0.9788
Logistic Regression	0.9543	0.9231	0.9384
Linear Regression	0.9714	0.9714	0.9714
Naive Bayes	0.9514	0.9714	0.9609

7 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

- cross validation iteration 1:
  Feature Set: feature generated with AttrEquivalenceBlocker implemented on "time" attributed followed by OverlapBlocker implemented on "title" attribute the performance is given in Figure 2.
- cross validation iteration 2:
  Feature Set: the feature implemented on the "time" and "title" attributes were remained the same, in addition we added a rule-based blocker with some data cleaning on "title" attribute, the performance is given in Figure 3.

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

Finally, after debugging on the matcher, we choose the random forest classifier as our best classifier.

- Precision: 98.14%
- Recall: 97.63%
- F_1 score: 97.88%

9 For each of the six learning methods, train the matcher based on that method on I, then report its precision/recall/F1 on J

We firstly trained the 6 models using set I and predicted results on set J, then we get the following performance on set J:

- **Decision Tree**
  Precision: 97.14% Recall: 89.14% F-1: 92.69%
- **SVM**
  Precision: 96.67% Recall: 81.84% F-1: 88.44%
- **Random Forest**
  Precision: 98.14% Recall: 97.63% F-1: 97.88%
- **Logistic Regression**
  Precision: 95.43% Recall: 92.31% F-1: 93.84%
- **Linear Regression**
  Precision: 97.14% Recall: 97.14% F-1: 97.14%
- **Naive Bayes**
  Precision: 95.14% Recall: 97.14% F-1: 96.06%

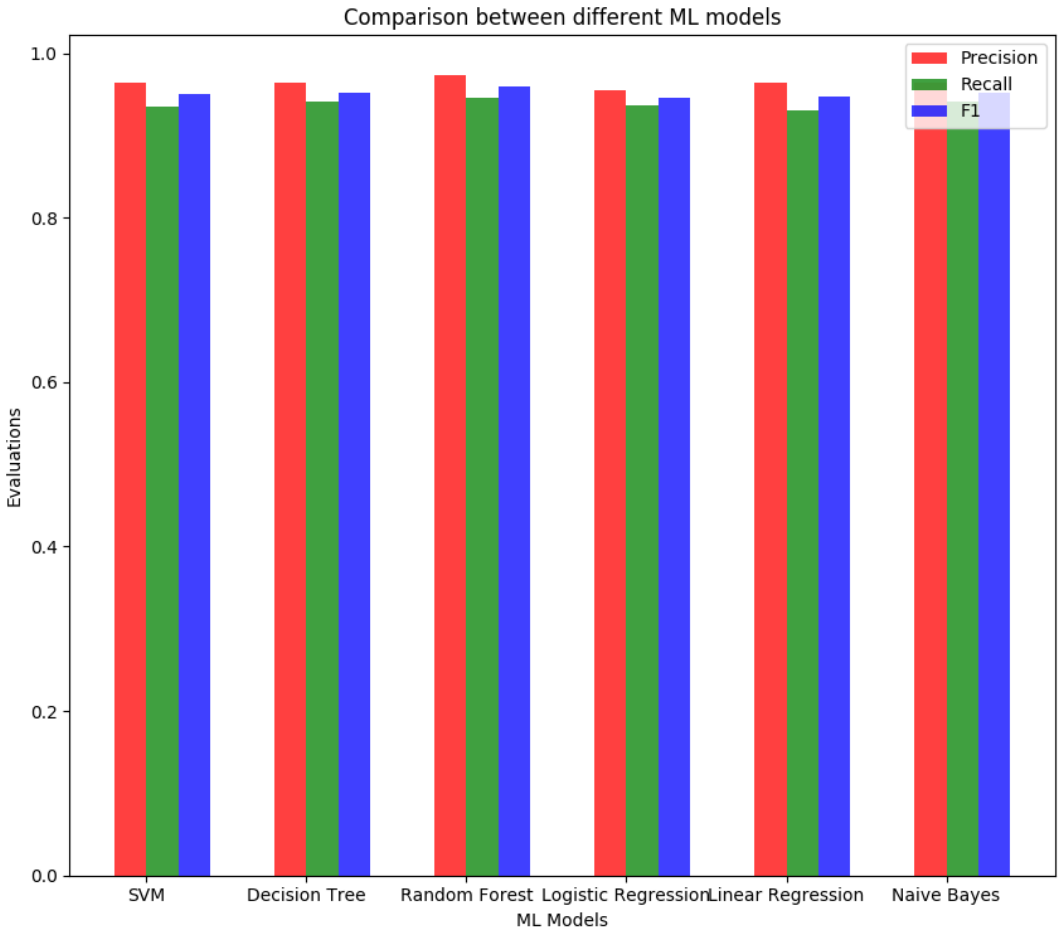

Figure 2: Matcher Performance Iteration 1

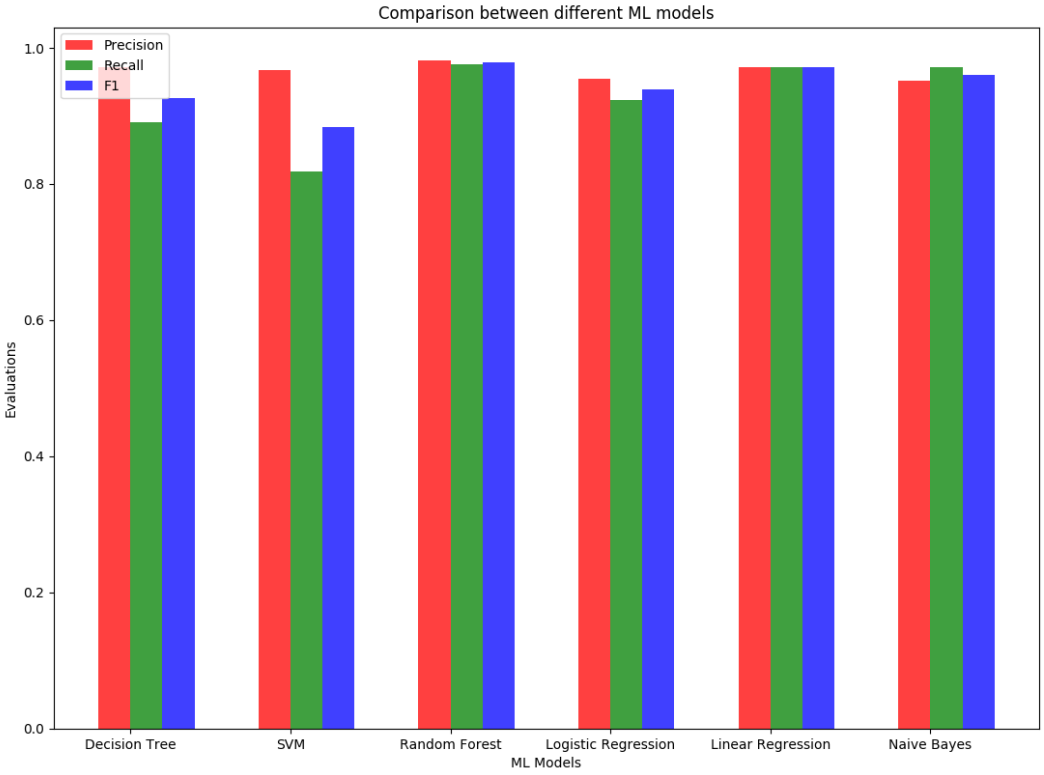

Figure 3: Matcher Performance Iteration 2

10 For the final best matcher Y selected, train it on I, then report its precision/recall/F-1 on J

The final best matcher selected is random forest. It result on J set:
Random Forest
Precision: 99.71%
Recall: 96.43%
F-1: 98.04%

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

- **Blocking**: Sample blocking run time took around 20 minutes, which EquivalenceBlocker took around 18 minutes, and the parallelized blocking took around 2 minutes.
- **Labelling Data**: Labeled manually, which took around 10 minutes to label 400 tuples.
- **Finding Best Matcher**: To generate the feature vectors and the first-round training and cross-validation, it took around 1 minute. For the debugging stage, it took around 30 minutes.

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

Recall is caused by relative large false negative numbers. We checked our output to find what factors caused the false negative. It turned out that almost all the false negative is due to the format of "title" (i.e. movies names) attribute are "noisy", which means the string format is not standardized and contained many irrelevant staff (e.g. Bfl O (ggGX /STwWcfl xZs 4). Thus, to further enhance the recall, we need to do more data cleaning with correspond to "movie titles".

13 Comments on Magellan

13.1 Positives

- Provide many effective methods similarity measure of strings. e.g. n-gram, which make it easy to get good performance
- Encapsulate many popular python scientific libraries.
- Provided good features that facilitate debugging

13.2 Negatives

- Down sampling has high time consumption, although the whole system get good performance
- Do down sampling with a quite large data size will lead to run out of memory

13.3 Expected Features

- We would like to see more machine learning models supported in Magellan, especially multi-layer neural network, even several deep learning models, like RNN, ResNet, and etc. Because, on earth these models result excellent performance in traditional ML literatures
- If possible we want to have some functions to visualize the ML learning performance like ROC curve, printing confusion matrix in multi-task learning.