Team Koala Ninja (Team 10)
Ankit Vij
Amanpreet Singh Saini
Joel Haynie

Class: CS 839 - Data Science Spring 2018
Prof: Dr. AnHai Doan
Date: 04/18/2018
RE: Project Stage 3

Entity Type: *Movies*

Table Info & Schema

We collected > 3000 tuples from each site across a variety of movie genres. We collected 3005 movies from rottenTomatoes and 3250 movies from imdb in stage 2. The Tuples are of the form:

```
ID, Title, Score, Rating, Genre, Directed By, Written By, Box Office, Release Date, Runtime, Studio
```

Stage 3 Pipline

Input (Stage #2) → Data Cleaning → Blocking → ML Match finder → Output (Matcher)

Data Cleaning

- Cleaned Stage #2 output: ImdbMovieDatabase.csv → A.csv & RottenTomatoesMovieDatabase.csv → B.csv
- Found and fixed some bad data fields
- Formated the data to be similar between the tables following the details below:
- Default values and empty values were filled in as labeled.
- We found issues in subsequent steps that forced us to refine and clean the data several times.

Name	Type	Default	Description	Cleaning
ID	String	NA	ID of this movie A_0000, B_0000	None
Title	String	NA	The title of the movie	None
Score	Number	0	The score given to the movie out of 0-100	Convert to Range
Rating	Categorical	NA	The rating assigned to the movie. Examples: PG-13, R, etc	Made Consistently Categorical
Genre	Categorical	NA	As single primary genre of the movie	Made Consistently Categorical
Directed By	String	NA	The primary director of the movie	Selected One
Written By	String	NA	The primary writer of the movie	Selected One
Box Office	Number	0	The amount of money made at the box office	Convert to Integer
Release Date	String	1900-1-1	The date of release of the movie; YYYY-MM-DD	Formatted Date
Runtime	Number	0	The duration of the movie in minutes	Convert to Integer

Studio	String	NA	The primary studio that made the Movie	Selected One

Blocking

Blocker(s) used and the number of tuple pairs in the candidate set obtained after the blocking step.

We used a combination of the attribute equivalence blocker, the overlap blocker and the rule-based blocker.

The attribute equivalence blocker:
First, we applied the attribute equivalence blocker on the attribute “Title” of the two tables A and B.
Second, we also applied the attribute equivalence blocker on the attribute “Release Date” of the two tables A and B. This was not done in sequence to the first result but independently on A and B.

The Overlap Blocker
Independent to the results of the above two blockers, we applied the overlap blocker on the “Title” attribute of A and B with the overlap size of 2. Since we are dealing with movies, the titles also contain some stop words like “Of”, “The”, etc. So, we remove these stop words while doing the blocking.

Combining Blockers
We then combine the results from the three methods above into an intermediate candidate set using the function `em.combine_blocker_outputs_via_union`

Rule-based Blocking
We then applied rule-based blocking on this intermediate candidate set to get the final candidate set. The rule defined is: If the Levenshtein similarity for the Title attribute is less than 0.4, block them.

The number of tuples in A X B (i.e the cartesian product) were **9766250**.
After applying this blocking scheme, we got Candidate Set Size of **3335**.

Number of labeled tuple pairs in the sample G.

From the candidate set C, we sampled 300 tuple pairs and got S which we labeled further to get G.
We labeled 300 tuple pairs in G, out of which 54 are labeled as 1 and the rest are 0.

Learning a Matcher

Precision, Recall, and F-1 that obtained for each of the six learning methods provided in Magellan (Decision Tree, Random Forest, SVM, Naive Bayes, Logistic Regression, Linear Regression), using cross validation for the first time on Set I (training)

Matcher	Average Precision	Average Recall	Average F1
Decision Tree	0.966667	0.966667	0.966667
SVM	0.971429	0.971429	0.969231
Random Forest	1.000000	0.966667	0.981818
Logistic Regression	0.950000	1.000000	0.971429
Linear Regression	1.000000	0.869524	0.926263
Naive Bayes	1.000000	0.926667	0.959596

Learning Based Matcher selected after cross validation on set I.

We selected the **Random Forest** as our matcher after cross validation as it had the highest F1 score.

Debugging iterations and cross validation iterations Performed.

We debugged SVM, Logistic Regression, and Decision Tree as they were the next contenders after Random Forest but we couldn't find any useful insights so we didn't follow with further debugging. Furthermore, most of our matchers had a precision > 96% and recall > 92%, with our best matcher having a precision of 100% and a recall or ~97%, which was another reason further debugging wasn't necessary.
Though, we had to clean some of the data before we could achieve these results. Those are described in the Data Cleaning section above.

Final best matcher selected, and its precision/recall/F1 on set I.

We selected Random Forest as our final best matcher and it had the following precision, recall, and F1

FInal Best Matcher	Precision	Recall	F1
Random Forest	1.000000	0.966667	0.981818

Precision, Recall, & F-1 on set J (testing) for the six learning methods trained matchers on set I (training)

Matcher	Precision	Recall	F1
Decision Tree	0.9167	1.0000	0.9565
SVM	0.8750	0.9545	0.9130
Random Forest	1.0000	1.0000	1.0000
Logistic Regression	0.8800	1.0000	0.9362
Linear Regression	1.0000	1.0000	1.0000
Naive Bayes	0.9167	1.0000	0.9565

Final best matcher trained on I, and its Precision, Recall, and F-1 on J.

Final Best Matcher	Precision	Recall	F1
Random Forest	1.0000	1.0000	1.0000

Recall Discussion

For our final best matcher we obtained a perfect recall of 100% on the test set J, which is an improvement on the recall achieved on the training set I (~97%). Although we achieved a perfect recall, we should keep in mind the following points-

- The percentage of positive examples are less
- Experiment with a different blocking strategy - We might want to add more rules (corresponding to the other attributes to the rule-based blocker) to reduce the skew between the positive and negative tuple pairs produced in the candidate set and consequently, the sampled set G. Presently, G contains 300 tuples and 54 of them are positive labels which means 18% of the sampled set G are positive.

Time estimates: (a) to do the blocking, (b) to label the data, (c) to find the best matcher

- Blocking: 7-8 hours
- Labing data: 1 hour
- Finding the best matcher: 4-5 hours

Conclusions & What we learned

- The EM tools are picky about missing data, thus data cleaning is a very important step in making the tools work well.

Magellan Library Feedback

- The library is very easy to use.
- The Documentation and Jupyter examples were very well done
- The Documentation around the "`RuleBasedBlocker -> add_rule -> conjunct_list`" and all the features generated for blocking could use expanding.
- Simple integration into pycharm was great.
- Some dependencies were found only when we ran certain functions. For Example: When using em.label_table() we found we needed PyQt5 only after we started running the code. We're not sure how dependencies are discovered in python so this may be normal.

Tools Used

- PyCharm v2017.3.3
- python v3.6
- Jupyter v1.0.0
- matplotlib v2.2.2
- sympy v1.1.1
- py_entitymatching v0.3.0
- scipy v1.0.1
- PyQt5 v5.10.1