Data Science - Project Stage 3
Entity Matching

Team Members:

- Karan Dharni (dharni@wisc.edu)
- Sudarshan Avish Maru (smaru@wisc.edu)
- Amogh Joshi (asjoshi4@wisc.edu)

Type of entity:

We used books as entity type for this project.
We extracted the data (in stage 2) from the following two sources:
1. www.amazon.com 2. www.booksamillion.com
In stage 2, we got 2990 tuples from Amazon and 9420 tuples from Booksamillion. However, on having a close look at the data, we felt the need to clean it (Ex: removing duplicates). We removed such tuples. After this step, we had **2903 tuples (Table A)** from Amazon and **9094 tuples (Table B)** from Booksamillion.

Blocking step:

We used the rule based blocker for this step. We also tried using attribute equivalence and overlap blockers, but many true matches were being blocked. For the rule based blocker, we used the following rules :
1. **Jaccard** similarity on **'Author'** attribute with threshold 0.2.
2. **Cosine** similarity on **'Name'** attribute with thresholds 0.3.
3. **Mel** similarity on **'Author'** attribute with threshold 0.5.

Using these, we were able to bring the total number of **tuple pairs (in Table C) down to 4490**. Then, using the debug blocker, Table D, we verified that we are not blocking true matches. We adjusted the thresholds so as to ensure that no true matches were blocked and size of Table C was also relatively small.

Sampling and labeling step:

We **sampled 500 tuple pairs** in Table S. Then we labeled those manually (Table G).

Matching step:

Results of performing CV on Training Set I for the first time:

Matcher	**Average Precision**	**Average Recall**	**Average F1**
Decision Tree	0.928421	0.924786	0.918968
Random Forest	**0.966013**	**0.924786**	**0.941175**
SVM	1.000000	0.199145	0.328333
Linear Regression	0.597421	0.838355	0.690269
Logistic Regression	0.896667	0.882692	0.885463
Naive Bayes	0.878788	0.969231	0.915246

We consider the results from **Random Forest** to be the best**.** As can be seen from the above table, we got approximately **96.60% precision, 92.48% recall and 94.12% F1 score.** As we got a precision greater than 90% with a very high recall, we did not feel the need to debug further.

After training on Set I, we got the following results **on Set J**:

Matcher	**Average Precision(%)**	**Average Recall(%)**	**Average F1(%)**
Decision Tree	96.36	98.15	97.25
Random Forest	96.00	88.89	92.31
SVM	88.89	29.63	44.44
Linear Regression	57.32	87.04	69.12
Logistic Regression	79.69	94.44	86.44
Naive Bayes	80.30	98.15	88.33

In this case, we got the best results with **Decision Tree**. As can be seen from the above table, we got **96.36% precision, 98.15% recall and 97.25% F1 score** with it. **Random Forest**, which was the best matcher for set I also gave good results with **96% precision, 88.89% recall and 92.31% F1 score.**

Time estimates:

a. Blocking step: **15** hours

b. Labeling the data: Approximately **2** hours

c. Finding the best matcher: **10** hours (Achieved the required precision with a high recall with little time spent on debugging).

Comments on Recall:

We achieved 92.48% recall (with Random Forest) on the training set I and 98.15% (with Decision Tree) on the test set J. To achieve even higher recall, we can focus on the blocking step. We can relax our constraints to avoid blocking of true matches. But, this has a trade-off of an increase in size of Table C. We tried to achieve this balance by varying the thresholds of applied rules and adding/removing rules. We analyzed the data closely to understand how the corresponding attributes of true matches in the 2 tables are similar. This helped us develop rules which gave us better results.

Comments on Magellan:

Good features:

1. Documentation is detailed and concise. Examples using Jupyter notebooks are very helpful.
2. Automated K-fold cross validation makes learning-based matchers easy to implement.

Possible improvements:

1. Installation process can be more streamlined. Instructions to install PyQt5 and XgBoost can be more detailed.
2. A visual representation of the complete entity-matching pipeline at the beginning of the documentation will be very helpful.
3. Support for distributed architecture will help speed up the blocking process (Eg: Blocking using Lev distance features is slow).
4. Ideally, all packages should support Python 2.7+. Currently, PyQt5 requires Python 3.