CS 839 Data Science Project Stage 2 Report

Sreejita Dutta, Deepanshu Gera, Rahul Jayan

Data Description

We have chosen books as the entity. We scraped this data from two web sources: walmart.com and amazon.com. The scraped table consists of the following attributes:

1. Name/Title of the book
2. Sale Price
3. Category/Genre
4. Author
5. ISBN
6. Pages
7. Publisher
8. Language
9. Dimension of the book
10. Weight of the book
11. User Rating

Though we have retained the attribute ISBN in the dataset, we have **not used ISBN** for blocking or entity matching.
amazon_books.csv consists of **3212 tuples** and walmart_books.csv consists of **3114 tuples**.

Blocking to generate candidate pairs

We have used three types of blockers for our blocking stage.

1. Black Box Blocker to handle single word 'Name'
2. Word level Overlap Blocker on attribute 'Author' with overlap size 2
3. Word level Overlap Blocker on attribute 'Name' with overlap size 2
4. Rule based blocker with absolute norm compute on attribute 'Pages'
5. Rule based blocker with absolute norm compute on attribute 'Sales_Price'

We obtained **802** tuple pairs after this blocking stage. We labelled **300** of these pairs for our learning stage.

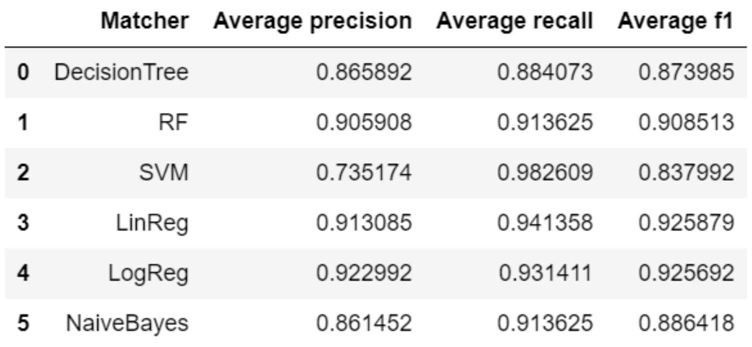
Sampling and labeling the candidate set

We labelled **300** of the blocked pairs for our learning stage.

Matching

I. **Initial Accuracy Metrics for development set**

   Iteration 1: The results obtained on performing cross validation for the first time on set I is compiled in the table given below.

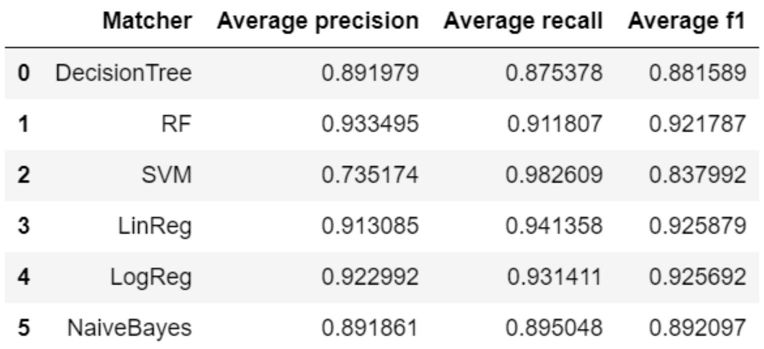


	Matcher	Average precision	Average recall	Average f1
0	DecisionTree	0.865892	0.884073	0.873985
1	RF	0.905908	0.913625	0.908513
2	SVM	0.735174	0.982609	0.837992
3	LinReg	0.913085	0.941358	0.925879
4	LogReg	0.922992	0.931411	0.925692
5	NaiveBayes	0.861452	0.913625	0.886418

We selected Random Forest as the matcher since it was giving optimal precision and recall. Also, since Magellan provides debugger support for only Decision Tree and Random Forest matchers, we suspected that after debugging RF could outperform Linear Regression and Logistic regression matchers.

II. **Debugging the classifier**

Iteration 2: We decided to debug RF matcher for this iteration. On debugging we realized that a lot of false positives and false negatives were due to words indicating the type of print in the 'Name' of the book, for e.g. Paperback, Hardcover etc. Current features do not account for this and adding a feature incorporating this difference in format can potentially improve the accuracy metrics. So, we added a blackbox feature to address this issue.
After this step, we obtained the following accuracy on the matchers.



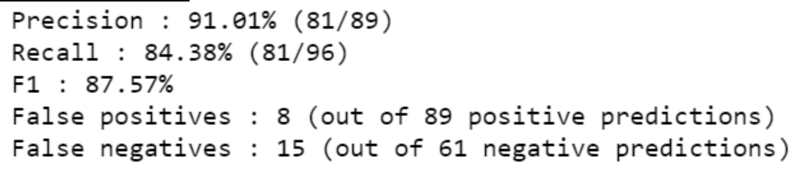
	Matcher	Average precision	Average recall	Average f1
0	DecisionTree	0.891979	0.875378	0.881589
1	RF	0.933495	0.911807	0.921787
2	SVM	0.735174	0.982609	0.837992
3	LinReg	0.913085	0.941358	0.925879
4	LogReg	0.922992	0.931411	0.925692
5	NaiveBayes	0.891861	0.895048	0.892097

From these results we selected Random Forests matcher as the best one as it was giving the highest precision and optimal recall.

III. **Accuracy Metrics for evaluation set**

We selected Random Forests as the best matcher and we ran it on test dataset J. The results obtained on test dataset J is given below:

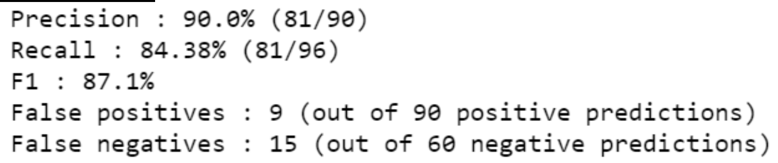
Random Forests



```
Precision : 91.01% (81/89)
Recall : 84.38% (81/96)
F1 : 87.57%
False positives : 8 (out of 89 positive predictions)
False negatives : 15 (out of 61 negative predictions)
```

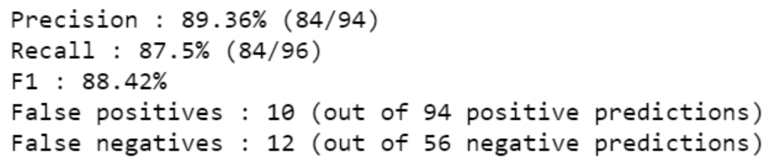
Result on test dataset for other 5 classifiers are given below:

Decision Tree



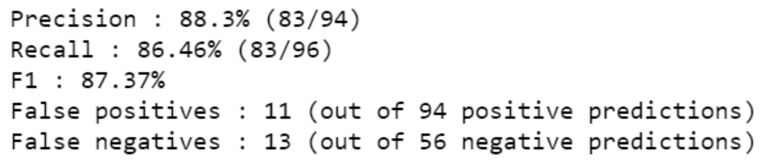
```
Precision : 90.0% (81/90)
Recall : 84.38% (81/96)
F1 : 87.1%
False positives : 9 (out of 90 positive predictions)
False negatives : 15 (out of 60 negative predictions)
```

Logistic Regression



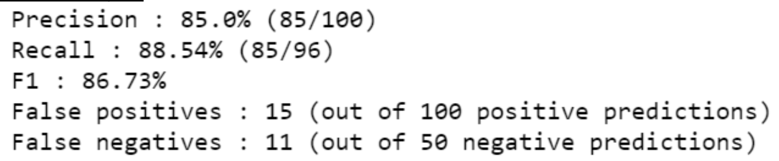
```
Precision : 89.36% (84/94)
Recall : 87.5% (84/96)
F1 : 88.42%
False positives : 10 (out of 94 positive predictions)
False negatives : 12 (out of 56 negative predictions)
```

Linear Regression



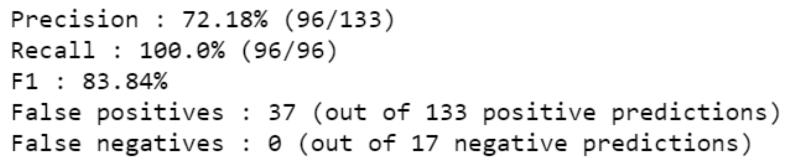
```
Precision : 88.3% (83/94)
Recall : 86.46% (83/96)
F1 : 87.37%
False positives : 11 (out of 94 positive predictions)
False negatives : 13 (out of 56 negative predictions)
```

Naïve Bayes



```
Precision : 85.0% (85/100)
Recall : 88.54% (85/96)
F1 : 86.73%
False positives : 15 (out of 100 positive predictions)
False negatives : 11 (out of 50 negative predictions)
```

SVM



```
Precision : 72.18% (96/133)
Recall : 100.0% (96/96)
F1 : 83.84%
False positives : 37 (out of 133 positive predictions)
False negatives : 0 (out of 17 negative predictions)
```

Random Forests gave the highest precision and recall on the evaluation set J

Time Estimates

1. Blocking: 4 hours
2. Labelling: 2 hours
3. Finding best matcher: 10 hours

Discussion

Matching books from the two web sources (amazon.com and walmart.com) was not straightforward. After looking at the dataset, we realized that books which had different names or authors or even ISBNs could represent the same entity. This makes it very hard for a classifier to distinguish a match from a mismatch for candidate pairs that look similar. For example, consider the candidate pairs:

- (“Sword Art Online, Vol. 2”, “Sword Art Online 10”)
- (“The Charm School Paperback”, “The Charm School”)

The first tuple pair is a mismatch, but the second tuple pair is a match. Since it is hard for the classifier to account for these minute differences, the second tuples pair is a false negative. Such cases lower the recall.

Magellan points

Positives

1. Good and clear documentation
2. Debugger for decision tree and random forests matcher is helpful

Potential improvements

1. Automatic feature extraction doesn’t seem to work for string attributes that vary in lengths. For Example, we had extracted book names from Walmart and Amazon. But, the attribute types that Magellan extracted was short (1 to 5 words) for Walmart books and short (5 to 10 words) for Amazon books. Because of this, the automatic feature extraction was not possible. We had to shorten the Amazon book names for Magellan to work.
2. Availability of debugging matchers for all learning-based algorithms would be helpful. It was extremely convenient to use the debuggers for Decision Tree and Random Forest. If there were debuggers for other types of matchers as well, it would really help in understanding the underlying problems.
3. GUI for labelling data could have option of hiding columns and persisting previously labelled columns. For Example, our tables had 11 columns each. Hence the table generated for labelling had approximately 22 columns, even though we were only interested in 8 of them for labelling. It would have been useful if we could just select the columns that we wanted to see while labelling the data.