Book Entity Matching Using Magellan
Deliverable 4

Majid Aksari, Dillon Skeehan

November 25, 2015

Abstract

In this project we complete all steps of entity matching process including crawling Web pages, information extraction, converting Web pages into structured data, and finally using a data matching system to block and match data. This report describes our experience using "Magellan" for matching the two book tables that we extracted from "Barnes and Nobel" and "Amazon" websites.

1 Matching using ISBN

1.1 Start

Initial performance of the six learning algorithms on the training data I from the sample S using just the page length feature.

Learning Method	Precision	Recall	F1
Decision Tree	77.2%	74.99%	80.37%
Random Forest	89.40%	77.47%	87.11%
Support Vector Machine	94.24%	72.88%	81.71%
Naive Bayes	59.59%	87.33%	72.2%
Log Regression	86.35%	73.30%	75.67%
Lin Regression	82.47%	70.19%	73.97%

After this initial cross validation, we selected the SVM learner given its consistent precision and recall. This would prove troublesome as Magellan didn't support built in methods for debugging a SVM learner and we didn't have previous experience working with SVM's.

1.2 Iterative Debugging

When trained on a sample U and tested on V, we achieved the following results:

Learning Method	Precision	Recall	F1
Support Vector Machine	81.8%	69.2%	75.0%

In order to improve precision, we then added a feature to compute the levenstein distance for titles. We found the following for cross validation:

Learning Method	Precision	Recall	F1
Decision Tree	61.5%	62.7%	47.1%
Random Forest	66.1%	73.8%	71.9%
Support Vector Machine	79.6%	75.2%	72.4%
Naive Bayes	73.4%	85.8%	78.3%
Log Regression	79.3%	52.1%	57.7%
Lin Regression	76.0%	74.8%	79.8%

Upon retraining on a new sample U and testing on a sample V, we didn't achieve any better performance:

Learning Method	Precision	Recall	F1
Support Vector Machine	81.8%	69.2%	75.0%

However, we did note that both the DT and RF learners improved with this change, but not to the performance level of the SVM learner. We then added another feature which broke down the authors of the book into 3-grams and used a jaccard similarity measure to try to improve the matching. We achieved the following results for cross validation.

Learning Method	Precision	Recall	F1
Decision Tree	55.5%	57.4%	44.7%
Random Forest	76.6%	49.1%	73.3%
Support Vector Machine	80.0%	78.1%	74.3%
Naive Bayes	77.7%	92.6%	77.6%
Log Regression	85.3%	50.6%	54.5%
Lin Regression	75.1%	73.45%	73.2%

1.3 Rules

At this point, we added an new rule to automatically add a tuple to the matching set if the ISBN's of the two book compared favorably. Thus our new rule would help improve precision, given that recall was was already quite high. In doing so we achieved the following results:

ISBN Rule	Precision	Recall	F1
Support Vector Machine	100.0%	86.67%	92.86%

1.4 Final Results

Each learning was trained on the training set I and then tested on the test set J. We achieved the following results:

Learning Method	Precision	Recall	F1
Decision Tree	68.75%	91.67%	78.57%
Random Forest	83.33%	83.33%	83.33%
Support Vector Machine	90.91%	83.33%	86.96%
Naive Bayes	57.89%	91.67%	70.97%
Log Regression	88.89%	66.67%	76.19%
Lin Regression	90.91%	83.33%	86.96%

We selected our best learning method Y as a SVM learner. Now with Y*, the SVM learner plus the rules we implemented was trained on I and tested on J. We achieved this final result:

Learning Method w/Rules	Precision	Recall	F1
Support Vector Machine	92.31%	100.0%	96.0%

2 Matching excluding ISBN

Since including ISBN could make matching easier, we tried matching without ISBN. We used as many features as possible, we added all the features generated for title, author, book cover, number of pages, price, and publication year. Here is our results from the first cross validation:

Learning Method	Precision	Recall	F1
Decision Tree	68.75%	91.67%	78.57%
Random Forest	83.33%	83.33%	83.33%
Support Vector Machine	90.91%	83.33%	86.96%
Naive Bayes	57.89%	91.67%	70.97%
Log Regression	88.89%	66.67%	76.19%
Lin Regression	90.91%	83.33%	86.96%

Linear regression, random forest, and svm all look good compared to others. Below you can find the results of all models on the test data J:

Learning Method	Precision	Recall	F1
Decision Tree	72.73%	76.19%	74.42%
Random Forest	76.47%	61.9%	68.42%
Support Vector Machine	83.33%	71.43%	76.92%
Naive Bayes	55.88%	90.48%	69.09%
Log Regression	88.89%	72.43%	75%
Lin Regression	90.91%	76.19%	82.05%

Linear regression actually performs very well on the test data with 90% precision and 76% recall, therefore, we choose it as our final classifier if we did not have access to ISBN. We plan to further debug this classifier for our final report.

3 Comments

- It would be helpful if I could see the function definitions, input, and output when I hovered over the function names. I had to look at the source code a few times to make sure about input and output.
- Generating features automatically sounds a good idea. However, we find the workflow for creating features manually not intuitive or at least not explained well.
- It is not clear if looking at the paths from random forest actually can help in debugging. What can we do about the paths? What kind of insight can we get?

- Combining the blocker outputs via union took a long time to run.
- We are not clear where the type of each attribute is specified. Magellan kept telling us that price from ltable has a different type from price from rtable and refused to automatically generate a feature for it.

4 Time Estimate

a) how much did it take to label the data? approximately 2 hour

b) to find the best learning-based matcher? approximately 25 hours

c) to add rules to the learning-based matcher? approximately 5 hours