CS 784 Project Stage 3 EXTENSION

Walter Cai {wcai@cs.wisc.edu}
Guangshan Chen {gchen9@wisc.edu}

December 14, 2015

1 First Round Mean Accuracy over Different Learning Methods on Training Data

When using all four attributes; `title, director(s), writer(s), actor(s)`:

Learner	Precision	Recall	F1
Decision Tree	0.987689	0.989994	0.988760
Random Forest	0.989118	0.994444	0.995000
SVM	0.968742	1.000000	0.980043
Naive Bayes	0.989408	0.979426	0.983678
Logistic Regression	0.974724	1.000000	0.986478
Linear Regression	0.978940	0.993939	0.977646

However, these preliminary results are too accurate to allow improvement in later debugging stages. Hence, we retrained the preliminary learners operating on only one of the 4 attributes at a time in the pursuit of finding less predictive power.

1. `title`

	Precision	Recall	F1
Decision Tree	0.98906099999999997	0.99459500000000001	0.99275100000000005
Random Forest	0.98898600000000003	1	0.99473699999999998
SVM	0.97863199999999995	1	0.99146999999999996
Naive Bayes	0.995	0.97221299999999999	0.98390200000000005
Logistic Regression	0.97841299999999998	1	0.98902199999999996
Linear Regression	0.98960899999999996	0.99375000000000002	0.99201300000000003

2. `director(s)`

	precison	recall	F1
Decision Tree	0.89634400000000003	0.94799900000000004	0.93353200000000003
Random Forest	0.89478999999999997	0.97063699999999997	0.92303000000000002
SVM	0.90114000000000005	0.96139600000000003	0.92968700000000004
Naive Bayes	0.89808399999999999	0.95683499999999999	0.92785499999999999
Logistic Regression	0.90039100000000005	0.96186300000000002	0.93018000000000001
Linear Regression	0.89952699999999997	0.96688600000000002	0.93185600000000002

3. `writer(s)`

	precison	recall	F1
Decision Tree	0.94556700000000005	0.96482500000000004	0.94765200000000005
Random Forest	0.96376200000000001	0.96745999999999999	0.96495200000000003
SVM	0.95346699999999995	0.95742400000000005	0.952735
Naive Bayes	0.956789	0.87929999999999997	0.91603500000000004
Logistic Regression	0.95182	0.95162899999999995	0.95482500000000003
Linear Regression	0.96695500000000001	0.95335300000000001	0.95968600000000004

4. `actor(s)`

	precison	recall	F1
Decision Tree	0.961009	0.93754899999999997	0.95513899999999996
Random Forest	0.95687999999999995	0.98223700000000003	0.95560299999999998
SVM	0.95774000000000004	0.98933099999999996	0.97341599999999995
Naive Bayes	0.96201099999999995	0.96115899999999999	0.96170599999999995
Logistic Regression	0.96049600000000002	0.98928799999999995	0.97233700000000001
Linear Regression	0.94993700000000003	0.97237200000000001	0.96429200000000004

We found that `director(s)` was the least predictive of the 4 so we will only use `director(s)`.

2 Learner Choice

Considering precision and recall, we elected to proceed with the **Logistic Regression** Learner.

3 Debugging

3.1 ITERATION 1

We attempt to improve the precision and recall on the logistic regression model. We split the training labeled dataset `I` into datasets `U` and `V`. In this training stage, We train the logistic regression model is trained on `U` and tested on `V`. Below is the evaluation on the predictions over dataset `V`:

```
Precision : 84.48% (49/58)
Recall : 94.23% (49/52)
F1 : 89.09%
False positives : 9 (out of 58 positive predictions)
False negatives : 3 (out of 26 negative predictions)
```

We attempt to debug the false positives. However, the false positives are distinct movies that share the exact same director set and hence cannot be improved with respect to the `director(s)`-only model.

3.2 Iteration 2

Hence, the only solution is to add rules on different attributes. In this case we opt to add a Jaccard similarity score rule on title which may be found in [Sec 5]. Below we will find the new evaluation on the predictions over dataset `V`:

```
Precision : 97.96% (48/49)
Recall : 92.31% (48/52)
F1 : 95.05%
False positives : 1 (out of 49 positive predictions)
False negatives : 4 (out of 35 negative predictions)
```

Note that precision has increased by over 13% while recall dropped by about 2%.

3.3 Iteration 3

We attempt to debug on false negatives. We discover one problem has to do with first-name last-name order swapping. We checked the similarity score for Jaccard on the following name variation:

```
Hsiao-Hsien Hou
Hou Hsiao-hsien
```

We believe this is symptomatic of a data cleaning problem (which we had missed earlier). We don't have a good method to fix this problem at this point besides hand cleaning the data.

Another problem is when one of multiple directors is omitted. For example:

```
Joel Cohen
Joel Cohen; Ethan Coen
```

This is another data cleaning type problem for which we would be forced to change manually.

3.4 Iteration 4

coffee break...

4 Final Learner and Associated Statistics using Cross Validation Iterations

Metric	Num folds	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean score
precision	5	1	1	1	1	0.958333	0.991667
recall	5	0.964286	0.923077	0.956522	0.923077	0.92	0.937392
f1	5	0.981818	0.96	0.977778	0.96	0.938776	0.963674

Note that recall has decreased from 0.961863 to 0.937392 while precision has increased from 0.900391 to 0.991667.

5 Manual Rules

```
# Add trigger - target false positives: use title related feature
neg_trigger = mg.MatchTrigger()
neg_trigger.add_cond_rule('title_title_jac_qgm_3_qgm_3(ltuple, rtuple) < 0.6', feat_table)
neg_trigger.add_cond_status(True)
neg_trigger.add_action(0)
```

6 Accuracy over Models Trained on Training Data and Applied to Test Data

6.1 Base Learners

Learner	Precision	Recall	F1
Decision Tree	0.961538	0.986842	0.974025
Random Forest	0.961538	0.986842	0.974025
SVM	0.962025	1.000000	0.980645
Naive Bayes	0.962025	1.000000	0.980645
Logistic Regression	0.962025	1.000000	0.980645
Linear Regression	0.962025	1.000000	0.980645

6.2 Final 'Best' Learner Y

[WITHOUT additional rule]

Learner	Precision	Recall	F1
Logistic Regression (without rules)	0.962025	1.000000	0.980645

6.3 Final 'Best' Learner Y*

[INCLUDING additional rule]

Learner	Precision	Recall	F1
Logistic Regression (including rules)	0.987013	1.000000	0.993464

7 Approximate Time Estimates

7.1 labeling

This ended up being our most time expensive task.

At the conclusion of stage 2, we were left with a candidate set of ∼1 million pairs down from a total possible of ∼9 million. Since there could be a maximum of ∼3000 matches to begin with drawing from the two 3000 tuple tables it was unlikely that we could randomly sample a set of 400 sample with a sufficient number of matches appearing at random. The first sampling+labeling attempt yielded a match likelihood of ∼0.002 (suggesting ∼2000 matches exist within the full candidate set).

We then decided to block further on the 1 million candidate pairs only for the purpose of generating a random golden label set. Our first blocker set the requirement that the `title` fields of a pair share minimum one token. This resulted in a candidate pair subset of size ∼100000. The second sampling+labeling attempt yielded a higher match likelihood but still not the necessary ≥50 matches from a random subset of size 400.

We further restricted that the `director` field share minimum one token. This resulted in a candidate pair subset of size ∼3000. From this it was possible to generate a golden label random subsample of size 400 with more equitable match and not-match partitions. The final partitioning for the golden labeling set was 257 positive and 143 negative.

One partner first traversed the list to generate initial labels, and then the second partner reviewed and edited those labelings.

We estimate this step took an additional ∼8 hours.

7.2 Finding the best Learning-Based Matcher

We estimate this step took an additional ∼8 hours.

7.3 Adding Custom Rules

We estimate this step took an additional ∼1 hours.