839 Project Stage 3: Matching Movie Entities from IMDB and Movie Numbers

Daniel K. Griffin
Department of Computer Science
University of Wisconsin Madison
dgriffin5@wisc.edu

Yudhister Satija
Department of Computer Science
University of Wisconsin Madison
ysatija@wisc.edu

Mitali Rawat
Department of Computer Science
University of Wisconsin Madison
mitali.rawat@wisc.edu

1 Introduction

In this project stage we performed entity matching between two data tables holding movie information. The sources that this data was collected from is provided below, and is the same as the data sources for project stage 2. Below, we provide a quicklist that can be used for quickly grading our assignment. We provide some further details in later sections.

2 Quick List

1. **Description of entity types:** We are matching movie entities in each table to one another. The entities have the attributes id, title, year, mpaa, runtime, genres, director, stars, and gross.
2. **Description of the two tables:** The two tables we use for matching are imdb3_neg_nan.csv and thenumbers3_neg_nan.csv. These tables have movie information that was originally scraped from IMDB.com and thenumbers.com. These tables are also special in that they have empty attributes filled in with a negative 1 if the attribute value was originally missing.
3. **The number of tuples per table:** Table1- imdb3_neg_nan.csv has 4291 tuples. Table2- thenumbers3_neg_nan.csv has 31006 tuples.
4. **Description of our Blockers:** We use 3 stages of overlap blockers, and 1 final custom overlap blocker that we ourselves implemented. The first blocker eliminates tuples that don't have an overlap in tokens for the movie title. This reduced out data set from 133,046,746 to 10,956,134 tuple pairs (about an order of magnitude). We then used another overlap blocker to block the data based on year. We first had to clean our data set to ensure that all data only had years and no extra characters. We also note that we tried using an attribute matching blocker, but it eliminated many positive tuples. The second blocker reduced our data from 10,956,134 pairs to 134,228 tuple pairs (about 2 orders of magnitude). Our third blocker was another overlap blocker that blocked based on mpaa rating. The total number of tuple pairs was reduced from 134,228 to 27,767. Our fourth and final blocker was a blocker that removed tuple pairs that were matched based on title only because they both shared the word "the". We implemented this blocker after debugging the output of the first two blockers and realized that many movies have the word "the" in the name. This blocker reduced the number of tuple pairs from 27,767 to 6,334, which was the final number of tuple pairs used for sampling and training our learning matchers.
5. **Number of tuple pairs in labeled sample G:** We labeled 600 tuple pairs in G.

6. **First time matchers precision and recall obtained:**
   (a) **Decision Tree:** Precision=0.992857, Recall=0.968190, F1=0.980080
   (b) **Random Forrest:** Precision=0.992857, Recall=0.984857, F1=0.988776
   (c) **SVM:** Precision=0.989474, Recall=0.747095, F1=0.848269
   (d) **Linear Regression:** Precision=0.992857, Recall=0.976857, F1=0.984524
   (e) **Logistic Regression:** Precision=0.993103, Recall=0.983333, F1=0.987796
   (f) **Naive Bayes:** Precision=0.992857, Recall=0.976857, F1=.0.984524
7. **Matcher selected**: Random Forest
8. **Report all debugging iterations and cross validation iterations that you performed:**
   (a) **Debugging Iteration 1:** Our first debugging iteration was focused on all of the matchers as a whole. When we initially ran our sampling and labeling stage, we sampled 600 instances, but only about 40 of them were true positives. So, when we performed cross validation and validation on the test set, our metrics were still high, but our test set was only 11 instances. So, we decided to implement a custom blocker removing instances that overlapped with only the word "the" (which was the main set of true negative instances in the sampled sets we were getting), and performed labeling on the resulting set, which gave us around 160 true instances out of the 160. After performing cross validation on this data set, we achieved the performance as above, but on a larger set of instances that were more varied in their nature.
9. **Final best matcher that you selected:** Random Forrest, Precision=0.992857, Recall=0.984857, F1=0.988776
10. **Final Matcher Metrics on J:**
   (a) **Decision Tree:** Precision=0.9565, Recall=0.9565, F1=0.9565
   (b) **Random Forrest:** Precision=1.0, Recall=1.0, F1=1.0
   (c) **SVM:** Precision=1.0, Recall=0.6957, F1=0.8205
   (d) **Linear Regression:** Precision=1.0, Recall=1.0, F1=1.0
   (e) **Logistic Regression:** Precision=1.0, Recall=1.0, F1=1.0
   (f) **Naive Bayes:** Precision=1.0, Recall=1.0, F1=1.0
11. **Final best matcher Y selected, trained on I and tested on J:** Random Forrest, Precision=1.0, Recall=1.0, F1=1.0
12. **Approximate time estimates:**
   (a) **Time to install software and learn to use it:** 3+ Hours
   (b) **Time to do the blocking:** 2 Hours
   (c) **Time to label the data:** 1 Hour
   (d) **Time to find the best matcher:** 1 Hour
13. **BONUS:** Refer to the "Bonus" section below. We kept a running list of problems we encountered and thoughts we had about Magellan in general.

3 Bonus: Notes on issues, errors, and suggestions

1. I used conda install -c anaconda pyqt to install the pyqt5 dependencies with anaconda 2 on MacOS Sierra.
2. You should have some test script that checks to ensure everything is working correctly after installation of all of the packages. Right now, you install everything, but you don't know if everything is working until you start trying to run code.
3. You might want to think about creating a "conda_install.sh" script that will install not only the py_entitymatcher, but also the necessary dependencies, rather than having a user download and install everything manually.
4. I would break the machine learning capability out of the py_entitymatcher tool, and focus the tool more on just the entity matching part, and feature generation. I think including the machine learning, while potentially beneficial to some people who don't know about ML (But if they use the tool, they probably know ML), it puts more to maintain in the

py_entitymatcher and somewhat locks down the matcher to future changes in the field of ML, ML models, and other ML package capabilities. I think that the tool should maintain the Unix style philosophy, which is do 1 thing, do it well, and make it easy to compose the tool with other tools (aka things like stdinput pipe handling, and output that is clean, etc...). The more compact this tool, the simpler it will be to be able to use and maintain it, and the more likely it will be that the industry adopts the tool. The simpler to install, use, and compose with other tools it is, the more likely the chances of widespread adoption will be.

5. Your links on this web page: http://anhaidgroup.github.io/py_entitymatching/v0.3.x/singlepage.html, for running ipy notebooks indicates that the link is a .ipynb file. But it is actually an html web page, and in that web page there is a link to download a .json ipy notebook. So this can be a bit confusing.

6. Your ipy notebooks are specific to the user "pradap" that created the notebook, and are not generalized. I ran into errors when running one of the notebooks bc of this.

7. You should make a docker file that can be run for this package.

8. Current anaconda compiled package for anaconda2 on MacOS Sierra "10.13.4" runs into issues with error: Traceback (most recent call last): File "<stdin>", line 1, in <module> File "/anaconda2/lib/python2.7/site-packages/py_entitymatching/__init__.py", line 42, in <module> from py_entitymatching.debugblocker.debugblocker import debug_blocker File "/anaconda2/lib/python2.7/site-packages/py_entitymatching/debugblocker/debugblocker.py", line 14, in <module> from py_entitymatching.debugblocker.debugblocker_cython import ImportError: dlopen(/anaconda2/lib/python2.7/site-packages/py_entitymatching/debugblocker/debugblocker_cython.so, 2): Symbol not found: __ZNSt11logic_errorC2EPKc Referenced from: /anaconda2/lib/python2.7/site-packages/py_entitymatching/debugblocker/debugblocker_cython.so Expected in: /usr/lib/libstdc++.6.0.9.dylib in /anaconda2/lib/python2.7/site-packages/py_entitymatching/debugblocker/debugblocker_cython.so

9. Error when installing using pip: "Cannot remove entries from nonexistent file /anaconda2/lib/python2.7/site-packages/easy-install.pth" Fixed by echoing an empty string to easy-install.pth file, and installing through pip without the "-U" option, which caused PyPrind-2.11.1 to be upgraded to PyPrind-2.11.2, but caused the installation error: Successfully installed PyPrind-2.11.2 backports.functools-lru-cache-1.5 kiwisolver-1.0.1 matplotlib-2.2.2 py-entitymatching-0.3.0 python-dateutil-2.7.2 pytz-2018.4 Traceback (most recent call last): File "/anaconda2/bin/pip", line 11, in <module> sys.exit(main()) File "/anaconda2/lib/python2.7/site-packages/pip/__init__.py", line 248, in main return command.main(cmd_args) File "/anaconda2/lib/python2.7/site-packages/pip/basecommand.py", line 252, in main pip_version_check(session) File "/anaconda2/lib/python2.7/site-packages/pip/utils/outdated.py", line 102, in pip_version_check installed_version = get_installed_version("pip") File "/anaconda2/lib/python2.7/site-packages/pip/utils/__init__.py", line 838, in get_installed_version working_set = pkg_resources.WorkingSet() File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 644, in __init__ self.add_entry(entry) File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 700, in add_entry for dist in find_distributions(entry, True): File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1949, in find_eggs_in_zip if metadata.has_metadata('PKG-INFO'): File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1463, in has_metadata return self.egg_info and self._has(self._fn(self.egg_info, name)) File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1823, in _has return zip_path in self.zipinfo or zip_path in self._index() File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1703, in zipinfo return self._zip_manifests.load(self.loader.archive) File "/anaconda2/lib/python2.7/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1643, in load mtime = os.stat(path).st_mtime OSError: [Errno 2] No such file or directory: '/anaconda2/lib/python2.7/site-packages/PyPrind-2.11.1-py2.7.egg'

10. Your ipynb files have some simple deprecation warnings.

11. In the ipynb guides, you should use "predictions[predictions.predicted == 1].head()" to display results of predictions so that people can examine the potentially matching entities to understand that the matching has actually occurred.
12. In the ipynb files, the notebooks say that the RF model has highest precision and recall, but I see the linear regression has highest precision and recall.
13. Giving an idea of the computational runtime of the debugger on some data set would be nice, or at least having a progress bar (like with the regular blocker) would be nice so that a user can gauge how long it will take to run the debugging phase. (We were running it initially with 10,956,134 tuple pairs)
14. The AttrEqBlocker doesn't seem to work for us on our tables when matching equivalence of years.
15. It's not clear to me how to load a list of labels from a file for the labeling portion. Manually labeling all of the data within the .ipynb isn't very practical for larger data sets, and it's not clear to me how to add labels to the data set in a manner that also generates ther required metadata.