CS 838 - Project Stage 4

Jale Dinler, Riccardo Mutschlechner, Steven Lamphear

dinler@wisc.edu, riccardo@cs.wisc.edu, slamphear@wisc.edu

April 16, 2017

- For project stage 3, before we started matching, we've used the 3 columns that were present in both data sets and ignored the other columns that were irrelevant to a matching. For this stage, to be fast and make our jobs easier while combining the two tables, we've constructed the tables again without omitting the irrelevant columns this time. We had 7682 possible matches after blocking last time. We applied the possible matchers to 3000 tuples sampled from possible matches and we've chosen random forest (99.4% precision, 99.3% recall). So we've applied RF to the all 7682 tuples this time, and we got 5337 matches.

  Last time when we were labeling the sampled data, we've noticed that there were missing values in year column in our pitchfork data set. But for discogs since our data set was big, we've cleared the data with missing year column. So when we were combining two tables we used the year column of discogs. We've also used the discogs data for artist and title name for consistency and then combine them with the rest of the columns.

- We have 5337 tuples in table E and 9 columns. Our table E looks like:

  p_id,d_id,artist,title,label,year,genres,score,best_new_music

  14183,221582,the wave pictures/the wave pictures,instant coffee baby,moshi moshi,2008,"rock,pop",7.8,0

  8784,198710,windsor for the derby/windsor for the derby,giving up the ghost,secretly canadian,2005,rock,6.8,0

  15387,304915,joan as police woman/joan as police woman,the deep field,pias,2011,rock,4.5,0

  2140,16976,dag nasty/dag nasty,minority of one,revelation,2002,rock,5.1,0

In [1]:
```
import pandas as pd
import numpy as np
```

In [52]:
```
df = pd.read_csv('out.csv', usecols=['id','title','artist','year','genres'])
df.id = df.id.drop_duplicates()
df = df[np.isfinite(df['id']) == True]
df = df[np.isfinite(df['year']) == True]
df.year = df.year.astype(int)
df = df[df['title'].astype(str) != " "]
df = df[df['artist'].astype(str) != " "]
df = df[(df['year'] >= 1999 ) == True]
df = pd.concat([df[col].astype(str).str.lower() for col in df.columns], axis=1)
df.to_csv('discog.csv', index=False, encoding = 'utf-8')
```

In [55]:
```
df = pd.read_csv('pitchfork_reviews.csv', usecols =['reviewid', 'title','artist'
,'year','label','score','best_new_music'])
df.reviewid = df.reviewid.drop_duplicates()
df = df[np.isfinite(df['reviewid']) == True]
df = pd.concat([df[col].astype(str).str.lower() for col in df.columns], axis=1)
df.to_csv('pitchfork.csv', index=False, encoding = 'utf-8')
```

In [57]:
```
import py_entitymatching as em
A = em.read_csv_metadata('pitchfork.csv', key='reviewid')
B = em.read_csv_metadata('discog.csv', key='id')
```

In [58]:
```
ob = em.OverlapBlocker()
```

In [66]:
```
C = ob.block_tables(A, B, 'title', 'title', word_level=True, overlap_size=3,
                    l_output_attrs=['title', 'artist', 'year','label','score','b
est_new_music'],
                    r_output_attrs=['title', 'artist', 'year','genres'],
                    show_progress=False)
```

In [67]:
```
ab = em.AttrEquivalenceBlocker()
D = ab.block_candset(C, 'title', 'title', show_progress=False)
D.to_csv('new_tuples_after_blocking.csv', index = False, encoding = 'utf-8')
```

In [119]:
```
df = pd.read_csv('possible_matches.csv', usecols = ['_id','ltable_reviewid','rt
able_id','ltable_artist','ltable_title','ltable_year','rtable_title','rtable_ar
tist','rtable_year','predicted'])
df = df[df['predicted'] == 1]
df = df.drop('predicted',1)
df = df.rename(columns = {'ltable_reviewid':'p_id','rtable_id': 'd_id','rtable_
artist' : 'd_artist', 'ltable_artist' : 'p_artist','ltable_title': 'p_title',
                    'ltable_year': 'p_year','rtable_title': 'd_title','rtable_
year': 'd_year'})
df.to_csv('matches.csv', index = False, encoding = 'utf-8')
```

In [80]:
```
p = pd.read_csv('new_tuples_after_blocking.csv')
```

In [111]:
```
p['predicted'] = df['predicted']
p = p[np.isfinite(p['predicted']) == True]
p = p.rename(columns={'ltable_reviewid':'p_id','rtable_id': 'd_id','ltable_labe
l': 'label','ltable_score': 'score', 'ltable_best_new_music': 'best_new_music',
'rtable_artist' : 'artist',
                      'rtable_title': 'title','rtable_year': 'year', 'rtable_gen
res':'genres'})
p = p[['p_id','d_id','artist', 'title', 'label', 'year', 'genres','score','best
_new_music']]
p.p_id = p.p_id.astype(int)
p.d_id = p.d_id.astype(int)
p.to_csv('table_E.csv', index = False, encoding = 'utf-8')
```

In [ ]:

In [2]:
```
import pandas as pd
import numpy as np
```

In [1]:
```
import py_entitymatching as em
A = em.read_csv_metadata('p_table.csv', key='reviewid')
B = em.read_csv_metadata('d_table.csv', key='id')
```

In [3]:
```
tmp = em.read_csv_metadata('tuples_after_blocking.csv',
                           key='_id',
                           ltable=A, rtable=B,
                           fk_ltable='ltable_reviewid', fk_rtable='rtable_id')
```

In [7]:
```
S = em.read_csv_metadata('labeled_tuples.csv',
                           key='_id',
                           ltable=A, rtable=B,
                           fk_ltable='ltable_reviewid', fk_rtable='rtable_id')
```

In [8]:
```
IJ = em.split_train_test(S, train_proportion=0.6, random_state=0)
I = IJ['train']
J = IJ['test']
```

In [9]:
```
F = em.get_features_for_matching(A, B)
```

In [10]:
```
H = em.extract_feature_vecs(I,
                            feature_table=F,
                            attrs_after='label',
                            show_progress=False)
```

In [11]:
```
any(pd.notnull(H))
```

Out[11]:
```
True
```

In [12]:
```
H = em.impute_table(H, exclude_attrs=['_id', 'ltable_reviewid', 'rtable_id', 'la
bel'], strategy='mean')
#K.to_csv('set_J.csv', index = False, encoding = 'utf-8')
```

In [13]:
```
rf = em.RFMatcher(name='RF', random_state=0)
```

In [14]:
```
rf.fit(table=H,
       exclude_attrs=['_id', 'ltable_reviewid', 'rtable_id', 'label'],
       target_attr='label')

# Convert J into a set of feature vectors using F
L = em.extract_feature_vecs(tmp, feature_table=F, show_progress=False)

# Predict on L
predictions = rf.predict(table=L, exclude_attrs=['_id', 'ltable_reviewid', 'rtab
le_id'],
              append=True, target_attr='predicted', inplace=False)
```

In [15]:
```
predictions.to_csv('predicted_matches.csv')
```

In [ ]:
```

```