Stage 4 - Entity Matching

4.1 Readin data and previous save matcher

In [1]:
```
from collections import Counter
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
import py_entitymatching as em
```

In [2]:
```
data_dir = './dataset/structured_data/'
A_filename = data_dir+'forbes_all_rename.csv'
B_filename = data_dir+'nasdaq_rename.csv'
blocked_res = data_dir+'blocking_results.csv'
shared_fields = ['Name',"Country", 'Industry', "MarketValue"] ## These
are shared fileds
# use the random forest matcher
m = em.load_object(data_dir+"rf_matcher.pkl"); # load matcher from pre
vious matching stage.
```

In [3]:
```
# Load orignial tables
A = pd.read_csv(A_filename, encoding = "ISO-8859-1" );
em.set_key(A, 'id');
A1 = A[['id'] + shared_fields];
em.set_key(A1, 'id');
B = pd.read_csv(B_filename,  encoding = "ISO-8859-1");
em.set_key(B, 'id');
B1 = B[['id'] + shared_fields];
em.set_key(B1, 'id');
# Load the pre-labeled data
S1 = em.read_csv_metadata(blocked_res,
                          key='_id',
                          ltable=A1, rtable=B1,
                          fk_ltable='ltable_id', fk_rtable='rtable_id',
encoding = "ISO-8859-1");
```

```
Metadata file is not present in the given path; proceeding to read t
he csv file.
```

4.2 Entity matching

Build features.

In [4]:
```
# Generate a set of features
F = em.get_features_for_matching(A1, B1)
# Add some new feature to F
def MarketValue_ratio(ltuple, rtuple) :
    try :
        return float(ltuple.MarketValue) / float(rtuple.MarketValue)
    except ValueError :
        return 0
em.add_blackbox_feature(F, 'MarketValue_ratio', MarketValue_ratio)
```

Out[4]:
```
True
```

In [5]:
```
F.feature_name
```

```
Out[5]: 0                                  id_id_exm
        1                                  id_id_anm
        2                             id_id_lev_dist
        3                              id_id_lev_sim
        4                  Name_Name_jac_qgm_3_qgm_3
        5              Name_Name_cos_dlm_dc0_dlm_dc0
        6              Name_Name_jac_dlm_dc0_dlm_dc0
        7                              Name_Name_mel
        8                         Name_Name_lev_dist
        9                          Name_Name_lev_sim
        10                             Name_Name_nmw
        11                              Name_Name_sw
        12           Country_Country_jac_qgm_3_qgm_3
        13       Country_Country_cos_dlm_dc0_dlm_dc0
        14       Country_Country_jac_dlm_dc0_dlm_dc0
        15                       Country_Country_mel
        16                  Country_Country_lev_dist
        17                   Country_Country_lev_sim
        18                       Country_Country_nmw
        19                        Country_Country_sw
        20         Industry_Industry_jac_qgm_3_qgm_3
        21     Industry_Industry_cos_dlm_dc0_dlm_dc0
        22     Industry_Industry_jac_dlm_dc0_dlm_dc0
        23                     Industry_Industry_mel
        24                Industry_Industry_lev_dist
        25                 Industry_Industry_lev_sim
        26                     Industry_Industry_nmw
        27                      Industry_Industry_sw
        28                         MarketValue_ratio
        Name: feature_name, dtype: object
```

Compute features for all blocked tuple pairs.

In [6]:
```
L1 = em.extract_feature_vecs(S1, feature_table=F, attrs_after='is_matc
h',show_progress=False)
L1 = em.impute_table(L1,
                    exclude_attrs=['_id', 'ltable_id', 'rtable_id',"is
_match"],
                    strategy='mean')
```

Predict using the matcher.

In [7]:
```
predictions = m.predict(table=L1, exclude_attrs=['_id', 'ltable_id', '
rtable_id',"is_match"],
                  append=True, target_attr='predicted', inplace=False)
```

In [8]:
```
predictions.to_csv(data_dir+"matching_result.csv") # save matching results to data directory
```

4.3 Find duplicates

In [9]:
```
import collections
fd = collections.defaultdict(list) # dictionary for forbes
nd = collections.defaultdict(list) # dictionary for nasdaq
lid = predictions["ltable_id"]
rid = predictions["rtable_id"]
match = predictions["predicted"]

# forbes dictionary
for i in range(len(lid)):
    if match[i] == 0:
        continue
    fd[lid[i]].append(rid[i])
    nd[rid[i]].append(lid[i])
```

For the multiple matches for one entity, we checked the results manunally and figured such similar entities are difficult to identify even by human beings. We printed one example below.

In [10]:
```
S1[198:256]
```

Out[10]:

	Unnamed: 0	_id	ltable_id	rtable_id	ltable_Country	ltable_Industry	ltable_Marke
198	4327	4327	1907	556	United States	Investment Services	60400
199	4344	4344	1907	557	United States	Investment Services	60400
200	4355	4355	1907	558	United States	Investment Services	60400
						Investment	

201	4366	4366	1907	559	United States	Services	60400
202	4371	4371	1907	560	United States	Investment Services	60400
203	4381	4381	1907	561	United States	Investment Services	60400
204	4415	4415	1907	562	United States	Investment Services	60400
205	4431	4431	1907	563	United States	Investment Services	60400
206	4438	4438	1907	564	United States	Investment Services	60400
207	4439	4439	1907	565	United States	Investment Services	60400
208	4440	4440	1907	566	United States	Investment Services	60400
209	4456	4456	1907	567	United States	Investment Services	60400
210	4471	4471	1907	568	United States	Investment Services	60400
						Investment	

211	4503	4503	1907	569	United States	Services	60400
212	4520	4520	1907	570	United States	Investment Services	60400
213	4537	4537	1907	571	United States	Investment Services	60400
214	4633	4633	1907	573	United States	Investment Services	60400
215	4648	4648	1907	574	United States	Investment Services	60400
216	4660	4660	1907	575	United States	Investment Services	60400
217	4678	4678	1907	576	United States	Investment Services	60400
218	4682	4682	1907	577	United States	Investment Services	60400
219	4692	4692	1907	578	United States	Investment Services	60400
						Investment	

220	4698	4698	1907	579	United States	Services	60400
221	4699	4699	1907	580	United States	Investment Services	60400
222	4710	4710	1907	581	United States	Investment Services	60400
223	4722	4722	1907	582	United States	Investment Services	60400
224	4742	4742	1907	583	United States	Investment Services	60400
225	4759	4759	1907	584	United States	Investment Services	60400
226	4770	4770	1907	585	United States	Investment Services	60400
227	4781	4781	1907	586	United States	Investment Services	60400
228	4799	4799	1907	587	United States	Investment Services	60400
229	4803	4803	1907	588	United States	Investment Services	60400
						Investment	

230	4804	4804	1907	589	United States	Services	60400
231	4805	4805	1907	590	United States	Investment Services	60400
232	4806	4806	1907	591	United States	Investment Services	60400
233	4820	4820	1907	592	United States	Investment Services	60400
234	4833	4833	1907	593	United States	Investment Services	60400
235	4848	4848	1907	594	United States	Investment Services	60400
236	4854	4854	1907	595	United States	Investment Services	60400
237	4855	4855	1907	596	United States	Investment Services	60400
238	4856	4856	1907	597	United States	Investment Services	60400
239	4857	4857	1907	598	United States	Investment Services	60400

240	4858	4858	1907	599	United States	Investment Services	60400
241	4859	4859	1907	600	United States	Investment Services	60400
242	4873	4873	1907	601	United States	Investment Services	60400
243	4889	4889	1907	602	United States	Investment Services	60400
244	4892	4892	1907	603	United States	Investment Services	60400
245	4903	4903	1907	604	United States	Investment Services	60400
246	4918	4918	1907	605	United States	Investment Services	60400
247	4924	4924	1907	606	United States	Investment Services	60400
248	4925	4925	1907	607	United States	Investment Services	60400

249	4926	4926	1907	608	United States	Investment Services	60400
250	4927	4927	1907	609	United States	Investment Services	60400
251	4949	4949	1907	610	United States	Investment Services	60400
252	4975	4975	1907	611	United States	Investment Services	60400
253	4997	4997	1907	612	United States	Investment Services	60400
254	5040	5040	1907	613	United States	Investment Services	60400
255	5065	5065	1907	614	United States	Investment Services	60400

Thus, we choose to only care about one-to-one matches, resulting in a total number of 704 matched entities.

In [11]:
```
## list of one-to-one matches
total_match = list() # list((l_id, r_id))

for ltable_id in fd :
    if len(fd[ltable_id]) == 1 :
        rtable_id = fd[ltable_id][0]
        if len(nd[rtable_id]) == 1 :
            total_match.append((ltable_id, rtable_id))

print (len(total_match))
```

```
703
```

4.4 New schema for combined table

First, we check the existing schema.

- Fileds shared with both tables

In [12]:
```
shared_fields
```

Out[12]:
```
['Name', 'Country', 'Industry', 'MarketValue']
```

- Fields only in table A (Forbes)

In [13]:
```
A_unique_fields = list(set(A.columns) - set(B.columns))
A_unique_fields
```

Out[13]:
```
['Assets', 'Employee', 'Sales', 'Profits']
```

- Fields only in table B (NASDAQ)

In [14]:
```
B_unique_fields = list(set(B.columns) - set(A.columns))
B_unique_fields
```

Out[14]:
```
['IPOyear', 'Symbol', 'LastSale', 'Summary Quote', 'Sector']
```

Based on the features in two tables, we define the new table schema to be the combine of all fields above, plus 'ltable_id' and 'rtable_id' as foreign key to original tables A and B.

In [15]:
```
combined_fields = shared_fields + A_unique_fields + B_unique_fields
E_fields = ['ltable_id', 'rtable_id'] + combined_fields
E_fields
```

Out[15]:
```
['ltable_id',
 'rtable_id',
 'Name',
 'Country',
 'Industry',
 'MarketValue',
 'Assets',
 'Employee',
 'Sales',
 'Profits',
 'IPOyear',
 'Symbol',
 'LastSale',
 'Summary Quote',
 'Sector']
```

4.5 Combining

Because NASDAQ table contains the company information formally registered on file, we use feature values from NASDAQ table if there is a conflict between two tables over shared fields (company name, industry, country).

In [16]:
```
A_fields = shared_fields + A_unique_fields
B_fields = shared_fields + B_unique_fields
```

In [17]:
```
A_indexed = A.set_index('id')
B_indexed = B.set_index('id')
```

In [36]:
```
E_match = pd.DataFrame(index=range(len(total_match)), columns=E_fields
)
for i, (ltable_id, rtable_id) in enumerate(total_match) :
    E_match.loc[i, 'ltable_id'] = ltable_id
    E_match.loc[i, 'rtable_id'] = rtable_id
    E_match.loc[i] = E_match.loc[i].combine_first(B_indexed.ix[rtable_
id][B_fields]).combine_first(A_indexed.ix[ltable_id][A_fields])

E_match.head()
```

Out[36]:

	ltable_id	rtable_id	Name	Country	Industry	MarketValue	Ass
0	2051	3485	Principal Financial Group Inc	United States	Accident &Health Insurance	18142.7	218
1	2053	3517	Prudential Financial, Inc.	United States	Life Insurance	45911.9	757
2	684	1103	Constellation Brands Inc	United States	Beverages (Production/Distribution)	31775.4	170
3	2061	4234	Torchmark Corporation	United States	Life Insurance	9082.82	199
4	19	372	Associated Banc-Corp	United States	Major Banks	3718.84	282

In [37]:
```
unique_ltable_ids = [i for i in A.id if i not in set(ltable_id for lta
ble_id, _ in total_match)]
E_A_only = pd.DataFrame(index=range(len(unique_ltable_ids)), columns=E
_fields)
for i, ltable_id in enumerate(unique_ltable_ids) :
    E_A_only.loc[i, 'ltable_id'] = ltable_id
    E_A_only.loc[i, A_fields] = A_indexed.ix[ltable_id][A_fields]

E_A_only.head()
```

Out[37]:

	ltable_id	rtable_id	Name	Country	Industry	MarketValue	Assets	Employee
0	1	NaN	77 Bank	Japan	Banks	1400	69100	NaN
1	2	NaN	Abu Dhabi Commercial Bank	United Arab Emirates	Banks	11000	62100	NaN
2	3	NaN	Abu Dhabi Islamic Bank	United Arab Emirates	Banks	3800	24300	NaN
3	4	NaN	Agricultural Bank of China	China	Banks	152700	2739800	NaN
4	5	NaN	Ahli United Bank	Bahrain	Banks	4200	34000	NaN

In [38]:
```
unique_rtable_ids = [i for i in B.id if i not in set(rtable_id for _,
rtable_id in total_match)]
E_B_only = pd.DataFrame(index=range(len(unique_rtable_ids)), columns=E
_fields)
for i, rtable_id in enumerate(unique_rtable_ids) :
    E_B_only.loc[i, 'rtable_id'] = rtable_id
    E_B_only.loc[i, B_fields] = B_indexed.ix[rtable_id][B_fields]

E_B_only.head()
```

Out[38]:

	ltable_id	rtable_id	Name	Country	Industry	MarketValue	Assets
0	NaN	1	1-800 FLOWERS.COM, Inc.	United States	Other Specialty Stores	665.526	NaN
1	NaN	2	1347 Property Insurance Holdings, Inc.	United States	Property-Casualty Insurers	48.8455	NaN
2	NaN	3	180 Degree Capital Corp.	United States	Finance/Investors Services	44.8111	NaN
3	NaN	4	1st Constitution Bancorp (NJ)	United States	Savings Institutions	148.506	NaN
4	NaN	5	1st Source Corporation	United States	Major Banks	1262.61	NaN

In [39]:
```
E = E_match.append(E_A_only).append(E_B_only).reset_index(drop=True)
E.tail()
```

Out[39]:

	ltable_id	rtable_id	Name	Country	Industry	MarketVa
7116	NaN	4710	ZTO Express (Cayman) Inc.	China	Trucking Freight/Courier Services	9574.11
7117	NaN	4711	Zumiez Inc.	United States	Clothing/Shoe/Accessory Stores	456.463
7118	NaN	4712	Zweig Fund, Inc. (The)	United States	NaN	180.265
7119	NaN	4713	Zynerba Pharmaceuticals, Inc.	United States	Major Pharmaceuticals	265.618
7120	NaN	4714	Zynga Inc.	United States	EDP Services	2474.38

In [40]:
```
E.to_csv(data_dir + 'E.csv')
```

4.6 Statistics of Table E

- Schema

In [41]:
```
E.columns
```

Out[41]:
```
Index(['ltable_id', 'rtable_id', 'Name', 'Country', 'Industry', 'Mar
ketValue',
       'Assets', 'Employee', 'Sales', 'Profits', 'IPOyear', 'Symbol'
,
       'LastSale', 'Summary Quote', 'Sector'],
      dtype='object')
```

- Number of tuples

In [42]:
```
len(E)
```

Out[42]:
```
7121
```

- Example tuples

1. Tuples from both tables A and B
2. Tuples from only table A
3. Tuples from only Table B

In [43]:
```
E.loc[[1, 563]]
```

Out[43]:

	ltable_id	rtable_id	Name	Country	Industry	MarketValue	Assets	Employee
1	2053	3517	Prudential Financial, Inc.	United States	Life Insurance	45911.9	757400	NaN
563	1508	1930	Golub Capital BDC, Inc.	United States	Food Markets	1097.56	NaN	20626

In [44]:
```
E.loc[[1000, 1234]]
```

Out[44]:

	ltable_id	rtable_id	Name	Country	Industry	MarketValue	Assets	Er
1000	316	NaN	China Biologic Products	Hong Kong	NaN	2941	NaN	Na
1234	588	NaN	Inditex	Spain	Apparel/Footwear Retail	103200	18800	Na

In [46]:
```
E.loc[[5000, 7000]]
```

Out[46]:

	ltable_id	rtable_id	Name	Country	Industry	MarketValue	Assets	E
5000	NaN	2217	Innoviva, Inc.	United States	Major Pharmaceuticals	1509.16	NaN	N
7000	NaN	4570	Western Asset Managed Municipals Fund, Inc.	United States	NaN	592.45	NaN	N

4.7 Merging function explained

We use NASDAQ as major table and Forbes as minor table. That is, we will proriorly use values from table B for shared fields if the field is present in both tables.

This can be best seen with those matches whose shared fields have N/A in both of the original tables A and B. We list match ids here.

In [23]:
```
for i_match, (ltable_id, rtable_id) in enumerate(total_match) :
    ltuple, rtuple = A_indexed.ix[ltable_id], B_indexed.ix[rtable_id]
    if any(ltuple[shared_fields].isnull()) and any(rtuple[shared_field
s].isnull()) :
        print(i_match)
```

```
563
```

Here is a pair of tuples from A and B.

In [24]:
```
i_match = 563
total_match[i_match]
```

Out[24]:
```
(1508, 1930)
```

In [25]:
```
ltable_id, rtable_id = total_match[i_match]
ltuple, rtuple = A_indexed.ix[ltable_id], B_indexed.ix[rtable_id]
```

In [26]:
```
ltuple.ix[A_fields]
```

Out[26]:
```
Name                   Golub
Country        United States
Industry        Food Markets
MarketValue              NaN
Assets                   NaN
Employee               20626
Sales                   3400
Profits                  NaN
Name: 1508, dtype: object
```

In [27]:
```
rtuple.ix[B_fields]
```

Out[27]:
```
Name                          Golub Capital BDC, Inc.
Country                                 United States
Industry                                          NaN
MarketValue                                   1097.56
IPOyear                                          2010
Symbol                                           GBDC
LastSale                                        19.87
Summary Quote    http://www.nasdaq.com/symbol/gbdc
Sector                                            NaN
Name: 1930, dtype: object
```

In [28]:
```
combined_tuple = pd.Series(index=E_fields)
```

In [29]:
```
combined_tuple = combined_tuple.combine_first(rtuple[B_fields])
combined_tuple
```

Out[29]:
```
Assets                                            NaN
Country                                 United States
Employee                                          NaN
IPOyear                                          2010
Industry                                          NaN
LastSale                                        19.87
MarketValue                                   1097.56
Name                          Golub Capital BDC, Inc.
Profits                                           NaN
Sales                                             NaN
Sector                                            NaN
Summary Quote    http://www.nasdaq.com/symbol/gbdc
Symbol                                           GBDC
ltable_id                                         NaN
rtable_id                                         NaN
dtype: object
```

In [30]:
```
combined_tuple = combined_tuple.combine_first(ltuple[A_fields])
combined_tuple
```

Out[30]:
```
Assets                                          NaN
Country                               United States
Employee                                      20626
IPOyear                                        2010
Industry                               Food Markets
LastSale                                      19.87
MarketValue                                 1097.56
Name                       Golub Capital BDC, Inc.
Profits                                         NaN
Sales                                          3400
Sector                                          NaN
Summary Quote    http://www.nasdaq.com/symbol/gbdc
Symbol                                         GBDC
ltable_id                                       NaN
rtable_id                                       NaN
dtype: object
```

Note that we use value from table A for "Industry" which only presents in A, value from table B for "MarketValue" which only presents in B, and value from table B for "Name" which presents in both A and B.

This is exactly what we got in E.

In [31]:
```
E_match.loc[i_match]
```

Out[31]:
```
ltable_id                                      1508
rtable_id                                      1930
Name                       Golub Capital BDC, Inc.
Country                               United States
Industry                               Food Markets
MarketValue                                 1097.56
Assets                                          NaN
Employee                                      20626
Sales                                          3400
Profits                                         NaN
IPOyear                                        2010
Symbol                                         GBDC
LastSale                                      19.87
Summary Quote    http://www.nasdaq.com/symbol/gbdc
Sector                                          NaN
Name: 563, dtype: object
```