Golden Notebook

- From this golden notebook, we are able to achieve a better performance of the five tasks. The *precision, recall and F1* can be found below. Compared to the performance of top-1 performer (see Table 6 in our paper), this golden notebook gets performance scores higher than (or in some tasks equal to) the top performer. ("*" denotes better performance)

Task	Precision	Recall	F1
Imprecise (P)	0.554*	0.860	0.674*
Inconsistent (C)	0.664	1.0	0.798
Duplicate (D)	1.0	1.0	1.0
Missing (M)	1.0	1.0	1.0
Non-unique (U)	1.0*	0.847	0.917*

0. Initialization of the dataset and working environment.

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

DB_FileName = "dataCleaningDB/formal_dataset_v1_181030.csv"
# DB_FileName = "dataCleaningDB/formal_dataset_simulation.csv"
db = pd.read_csv(filepath_or_buffer = DB_FileName)
```

1. Imprecise (P)

In [2]:
```
# From P2 (same result with experiment dataset as P14 below)
X = pd.to_datetime(db[db["join date"].notnull()]["join date"], errors='coerce')
str(list(X.loc[X.isnull()].index.values))
# pd.to_datetime(db[db["join date"].notnull()]["join date"], errors='coerce')
```

Out[2]:
```
'[0, 2, 3, 19, 21, 26, 27, 29, 30, 34, 35, 36, 37, 38, 47, 48, 66, 75, 78, 79,
81, 91, 94, 99, 108, 109, 110, 111, 113, 120, 121, 122, 132, 134, 136, 138, 15
5, 156, 160, 165, 170, 183, 184, 186, 202, 203, 210, 211, 212, 219, 220, 230,
231, 233, 238, 253, 262, 278, 285, 287, 288, 302, 304, 311, 312, 315, 317, 32
3, 324, 346, 353, 362, 364, 376, 383, 410, 411, 412, 413, 414, 417, 418, 419,
425, 428, 433, 441, 450, 461, 462, 465, 469, 474, 475, 477, 478, 479, 480, 49
0, 491, 497, 502, 519, 520, 521, 522, 532, 536, 537, 538, 541, 551, 556, 557
```

In [3]:
```
# From P14 (same result with experiment dataset as P2 above)
import re
cid_list = list()
inprecise_list = list()
missing = list()
regexp = re.compile(r'[0-9]{4}\/[0-9]{2}\/$')
null_list = pd.isnull(db['join date'])
for index, item in enumerate(null_list):
    if item:
        missing.append(index)
for index, cid in enumerate(db['join date']):
    if index in missing:
        continue
    cid = str(cid)
    if regexp.search(cid):
        inprecise_list.append(index)
print inprecise_list
```

```
[0, 2, 3, 19, 21, 26, 27, 29, 30, 34, 35, 36, 37, 38, 47, 48, 66, 75, 78, 79,
81, 91, 94, 99, 108, 109, 110, 111, 113, 120, 121, 122, 132, 134, 136, 138, 15
5, 156, 160, 165, 170, 183, 184, 186, 202, 203, 210, 211, 212, 219, 220, 230,
231, 233, 238, 253, 262, 278, 285, 287, 288, 302, 304, 311, 312, 315, 317, 32
3, 324, 346, 353, 362, 364, 376, 383, 410, 411, 412, 413, 414, 417, 418, 419,
425, 428, 433, 441, 450, 461, 462, 465, 469, 474, 475, 477, 478, 479, 480, 49
0, 491, 497, 502, 519, 520, 521, 522, 532, 536, 537, 538, 541, 551, 556, 557
```

In [4]:
```
# From P58
def get_index_str_from_df(df):
    return ",".join([str(val) for val in df.index])
contacts_region_code_9 = db[db["contact"].str.contains("^\+\d\d\d{9}", na=False
)]  # A
contacts_region_code_no_plus = db[db["contact"].str.contains("^\d\d\d{9}", na=F
alse)]  # B
contacts_landline = db[db["contact"].str.contains("^0\d{9}", na=False)]  # C
get_index_str_from_df(db[~db.index.isin(pd.concat([contacts_landline, contacts_
region_code_9, contacts_region_code_no_plus]).index)])
```

Out[4]:
```
'1,2,4,5,7,8,9,10,12,13,14,15,16,17,18,19,22,23,24,25,26,27,29,30,32,33,34,35,
36,37,38,39,40,41,42,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,
63,66,67,68,70,71,72,73,74,75,76,77,78,79,80,82,83,86,89,90,91,92,94,98,99,10
0,104,105,106,107,110,111,112,113,116,117,118,119,122,123,124,125,126,127,128,
129,132,134,135,136,137,139,140,141,143,144,145,146,147,148,150,151,153,154,15
5,156,158,159,160,162,163,164,165,166,167,168,169,171,174,175,176,177,178,179,
180,181,183,184,185,186,187,188,191,192,195,196,197,198,199,200,203,204,205,20
```

2. Duplicate (D)

In [5]:
```
# From P8  (label duplicated instances, label N-1 if there are N duplicates. no
t the same as P3 below)
lst=[]
for i, j in enumerate(db.duplicated().tolist()):
    if not j:
        continue
    lst.append(i)
print(lst)
```

```
[8, 16, 23, 27, 30, 33, 35, 37, 38, 40, 42, 45, 51, 56, 60, 62, 65, 68, 72, 7
4, 83, 85, 97, 103, 106, 109, 111, 115, 117, 121, 126, 129, 131, 140, 145, 14
7, 151, 156, 163, 173, 176, 177, 190, 192, 194, 197, 208, 211, 212, 215, 216,
220, 223, 231, 245, 260, 264, 271, 274, 275, 282, 283, 288, 294, 303, 304, 30
8, 312, 320, 324, 328, 330, 334, 336, 339, 341, 345, 349, 351, 356, 360, 367,
370, 375, 381, 382, 393, 394, 401, 404, 407, 413, 418, 419, 421, 424, 432, 44
5, 446, 452, 454, 467, 472, 475, 479, 480, 487, 491, 496, 500, 510, 517, 520
```

In [6]:
```
# From P3 (label ALL duplicate instances, label N if there are N duplicates)
print(db.loc[db.duplicated(keep=False)].index.tolist())
```

```
[7, 8, 14, 16, 22, 23, 26, 27, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 44, 45, 50, 51, 55, 56, 59, 60, 61, 62, 64, 65, 67, 68, 70, 72, 73, 74, 8
2, 83, 84, 85, 96, 97, 102, 103, 105, 106, 108, 109, 110, 111, 114, 115, 116,
117, 120, 121, 125, 126, 128, 129, 130, 131, 139, 140, 144, 145, 146, 147, 15
0, 151, 155, 156, 162, 163, 172, 173, 174, 175, 176, 177, 189, 190, 191, 192,
193, 194, 195, 197, 207, 208, 210, 211, 212, 214, 215, 216, 219, 220, 222, 22
3, 230, 231, 244, 245, 259, 260, 263, 264, 270, 271, 273, 274, 275, 280, 281
```

3. Non-Unique (U)

In [7]:
```
# From P6
A = db.loc[db.duplicated(['customer_id'], keep = False)].index.tolist()  # non-
Unique and Duplicate (recall=1.0)

"""
B = db.loc[db.duplicated(keep = False)].index.tolist()  # Duplicate
C = []
for x in A:
    if (x not in B):
        C.append(x)
#"""


# From P46
dbCount = db['customer_id'].value_counts()
valueList = dbCount.loc[(dbCount.values > 1)].index.tolist()
dbCount2 = db.loc[db['customer_id'].isin(valueList)]
dbCount3 = dbCount2['name'].value_counts()
valueList = dbCount3.loc[(dbCount3.values == 1)].index.tolist()
B = db.loc[db['name'].isin(valueList)].index.tolist()  # real experiment, preci
sion=0.951
C = []
for x in A:
    if (x in B):
        C.append(x)
print C
# print dbCount2.loc[dbCount2['name'].isin(valueList)].index.tolist()  # correc
t one, precision=1.0, recall=0.847
```

```
[1, 2, 5, 6, 9, 10, 11, 15, 17, 18, 46, 47, 57, 66, 71, 75, 76, 78, 79, 80, 8
1, 86, 87, 98, 99, 104, 113, 118, 119, 122, 124, 132, 133, 142, 143, 148, 157,
158, 161, 164, 165, 168, 169, 170, 178, 179, 181, 182, 184, 185, 187, 188, 21
3, 217, 218, 233, 234, 237, 238, 240, 246, 247, 252, 253, 255, 256, 257, 258,
265, 266, 268, 269, 276, 277, 306, 309, 310, 313, 319, 327, 340, 352, 353, 35
5, 361, 365, 371, 372, 376, 377, 383, 384, 391, 396, 397, 402, 409, 410, 423,
429, 430, 433, 434, 436, 437, 440, 441, 448, 449, 455, 456, 459, 460, 468, 46
```

4. Missing (M)

In [8]:
```
# From P9 (1st part out put 2 parts)
contact_nan = db.loc[(db['contact'].isnull())].index.tolist()
contact_nan_int = [int(v) for v in contact_nan]
print contact_nan_int
```

```
[44, 45, 154, 167, 229, 250, 269, 280, 283, 377, 423, 450, 468, 501, 509, 510,
521, 522, 523, 535, 543, 544, 548, 554, 636, 663, 704, 705, 709, 712, 725, 73
1, 733, 740, 756, 787, 808, 809, 833, 903, 904, 939, 941, 977, 987, 988, 1073,
1074, 1125, 1164, 1183, 1186, 1188, 1198, 1215, 1226, 1245, 1254, 1255, 1266,
1267, 1268, 1269, 1286, 1336, 1355, 1382, 1418, 1436, 1437, 1439, 1448, 1449,
1458, 1501, 1515, 1548, 1588, 1593, 1627, 1628, 1645, 1666, 1667, 1727, 1803,
1804, 1838, 1862, 1930, 1946, 1947, 1948, 1950, 1951, 1971, 1993, 2001, 2100,
```

In [9]:
```
# From P9 (2nd part out put 2 parts)
join_nan = db.loc[(db['join date'].isnull())].index.tolist()
join_nan_int = [int(v) for v in join_nan]
print join_nan_int
```

```
[11, 93, 149, 167, 172, 173, 191, 192, 195, 197, 225, 237, 242, 244, 245, 263,
264, 276, 295, 296, 300, 355, 380, 381, 382, 391, 463, 488, 540, 571, 574, 57
8, 584, 609, 621, 622, 624, 638, 643, 696, 710, 725, 756, 759, 785, 833, 845,
846, 883, 887, 906, 907, 941, 946, 947, 987, 988, 1007, 1014, 1034, 1036, 103
9, 1040, 1041, 1073, 1074, 1092, 1148, 1164, 1170, 1179, 1184, 1185, 1192, 123
3, 1340, 1341, 1356, 1357, 1365, 1405, 1491, 1497, 1506, 1549, 1581, 1672, 167
3, 1695, 1697, 1728, 1762, 1763, 1770, 1780, 1893, 1899, 1937, 1963, 1964, 198
```

In [10]:
```
# From P8 (not errors for these two columns)
print db.loc[(db['name'].isnull())]
print db.loc[(db['customer_id'].isnull())]
```

```
Empty DataFrame
Columns: [customer_id, name, contact, join date]
Index: []
Empty DataFrame
Columns: [customer_id, name, contact, join date]
Index: []
```

5. Inconsistent (C)

In [11]:
```
# From P2 (part 1/3)
X = db.loc[(db['join date'].notnull())]
Y = X.loc[(X['join date'].str.contains('^[0-9]+/[0-9]+/[0-9]+$', regex=True))]
str(list(X.loc[~X["join date"].isin(Y["join date"])].index.values))
```

Out[11]:
```
'[0, 2, 3, 4, 5, 7, 8, 9, 10, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 46, 47, 4
8, 49, 50, 51, 52, 53, 54, 57, 58, 59, 60, 63, 64, 65, 66, 67, 68, 69, 70, 72,
73, 74, 75, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 91, 92, 94, 98, 99, 10
2, 103, 104, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 120,
121, 122, 124, 127, 128, 129, 130, 131, 132, 133, 134, 136, 137, 138, 141, 14
2, 144, 145, 146, 147, 148, 150, 151, 152, 153, 154, 155, 156, 159, 160, 161,
```

In [12]:
```
# From P2 (part 2/3)
X = db.loc[(db['name'].notnull())]
Y = X.loc[(X['name'].str.contains('^[a-zA-Z ]+,[a-zA-Z ]+$', regex=True))]
str(list(X.loc[X["name"].isin(Y["name"])].index.values))
```

Out[12]:
```
'[7, 8, 21, 34, 35, 53, 61, 62, 69, 76, 77, 78, 81, 82, 83, 91, 94, 96, 97, 10
7, 116, 117, 135, 136, 138, 146, 147, 161, 164, 165, 169, 186, 187, 191, 192,
195, 197, 202, 203, 209, 210, 211, 212, 214, 215, 216, 217, 219, 220, 234, 23
6, 239, 241, 243, 248, 261, 262, 263, 264, 268, 280, 283, 290, 292, 295, 296,
298, 301, 302, 303, 304, 311, 312, 316, 325, 342, 346, 350, 351, 359, 360, 36
8, 385, 388, 392, 393, 394, 395, 399, 402, 403, 404, 405, 410, 412, 413, 416,
423, 429, 433, 439, 440, 442, 443, 448, 463, 478, 479, 480, 481, 495, 502, 51
```

In [13]:
```
# From P2 (part 3/3)
X = db.loc[(db['contact'].notnull())]
Y = X.loc[(X['contact'].str.contains('^[0-9]{10}$', regex=True))]
str(list(X.loc[~X["contact"].isin(Y["contact"])].index.values))
```

Out[13]:
```
'[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 2
3, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 46,
47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 66, 67, 6
8, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 12
2, 123, 124, 125, 126, 127, 128, 129, 132, 133, 134, 135, 136, 137, 139, 140,
```