CS838 Data Science
Course Project Stage Four

Group 7
Hongyi Wang, Hao Fu, Miao Yang

April 16, 2017

1 How did you combine the two tables A and B to obtain E?

In this stage, to create schema of table E, we looked into the data tables involve house conditions and selling prices crawled from **Zillow** instead of the data tables we used in stage 3. Because if we are going to merge "movies.csv" and "track.csv" we used in stage 3, we can only use two attributes, i.e. "title (i.e. movie names)" and "year" to do the data merging for this stage, which can be constraints for this stage if we want to build complex rules.
In this stage, the schema of table A is like:

$$\{tuple\_id\_A, \textbf{name}, \textbf{phone}, \textbf{year}, price, size\}$$

,in which name represents the names of the agents, phone number represents their phone numbers, year stands for in which year the house was built.
In table B, the schema is like:

$$\{tuple\_id\_B, \textbf{name}, \textbf{phone}, \textbf{year}, zip, type\}$$

Different from table A, the zip attribute in table B represents zip of the house for sale, while type attribute represent the type of the hose. In this project, the feature vector space of type is like: {condo, single family, multi family, town house}.
One can clearly see that, the attributes with name of bold type are attributes, which table A and table B have in common. We therefore use these attributes to do the data matching in this stage. The final schema of table E is like:

$$\{\textbf{name}, \textbf{phone}, \textbf{year}, price, size, zip, type\}$$

Note: the attribute "tuple_id_A", and "tuple_id_B" are only used for searching the original data tuples in table A and B based on the matches. Therefore, in schema E, we didn't show an attribute to describe the index of tuple.
A script in Python was used for this stage to implement the data merging rules, we will discuss detailed information about the data merging rules.

2 Did you add any other table?

We didn't add any other table for this stage.

3 When you did the combination, did you run into any issues?

We did encountered several issues when we did the combination.

- **out liar values in year attribute**: there is some out liar values for year (i.e. the year the house was built) like 2102, 9192, and etc. Our solution is to look the "year" value in the other table, if that one is reasonable, then we just choose that one for table E, if "year" values in both tables are out liars we just remove this tuple from table E (fortunately, we never faced this condition).

- **missing values**: sometimes we faced missing values for some attribute, similarly, our solution is to look into the corresponding tuple and attribute in the other table. If the corresponding value in the other is not missing, we just use that one. If both values are missing, we just leave the value blank.

4 Discuss the combination process in detail, e.g., when you merge tuples, what are the merging functions (such as to merge two age values, always select the age value from the tuple from Table A, unless this value is missing in which case we select the value from the tuple in Table B)

4.1 For attribute "name"

After data matching the names(name of human) are overlap to some extent, but basically they are still in different formats e.g. $\{first\ name{+}last\ name\}$, $\{first\ name{+}abbreviate\ of\ middle\ name + last\ name\}$, $\{first\ name + full\ middle\ name + last\ name\}$, and etc. We always assume that longer string will be more likely to provide complete information, thus here comes our rule to combine human name.
We first convert the whole name string to lowercase, then split it into substrings of "first name", "middle name", and "last name". Then for each matches, we compare these substrings correspondingly between two tuples, selecting each longer substrings and concatenate again to form the whole name and write into table E.

Examples:

- **Table** A: Nina Chen Landes, **Table** B: Nina C. Landes, **Table** E: Nina Chen Landes
- **Table** A: Michael Buckman, **Table** B: Mike C. Buckman, **Table** E: Michael C. Buckman

4.2 For attribute "year"

As we discussed in section 3, to avoid out liar values in year attribute. We always choose the year value in reasonable range. We assume there is no building that was built before 1800 are able for sale. Thus our range will be [1800, this year]. If neither values in A, and B are out liars, we just compare if they are the same value, if so we just choose either values, if not, we choose the value with lower values.

Examples:

- **Table** A: 2105, **Table** B: 1993, **Table** E: 1993
- **Table** A: 1985, **Table** B: 1997, **Table** E: 1985

4.3 For attribute "phone number"

Similar to attribute of name, the formats of attribute "phone number" are somehow different. e.g. {(XXX)-XXXX-XXXX}, {XXX-XXXX-XXXX}, or {XXX XXXX XXXX}. Thus, we first extract the numbers (without "-" and "()") in the phone number strings from table A, or B. Then, format it into the format of (XXX)-XXXX-XXXX.

Examples:

- **Table** A: 202-499-2547, **Table** B: 202 499 2547, **Table** E: (202)-499-2547
- **Table** A: (703)-782-8166, **Table** B: 703-782-8166, **Table** E: (703)-782-8166

4.4 For attributes not in common

Since we do not need to do any merging for those attributes that are not common in table A, B, we just write them into table E based on schema of E.

5 Statistics on Table E: specifically, what is the schema of Table E, how many tuples are in Table E? Give at least four sample tuples from Table E

- Final schema of table E: As we mentioned in section 1, the final schema of table is like:

$$\{\textbf{name}, \textbf{phone}, \textbf{year}, price, size, zip, type\}$$

- Number of tuples in table E: Finally, we have 375 tuples in table E.
- we show 10 examples of tuples in table E here:

william marry raveis, (888)699-8876, 1955, 379900, 6055,22314, single family
ricky b. schwartz, (781)850-4334, 2017, 468200, 2100, 22305, condo
jeffrey chubb, (617)299-8866, 1953, 152900, 590,21189, condo
bill kevin thompson, (774)901-5417, 1971, 279900, 1412, 23021, single family
richy i. jordan, (617)936-7302, 2015, 388995, 1325, 21190, condo
lamacchia ruby king, (844)201-0842, 1955, 249900, 1020, 23032, single family
beacon rock, (617)285-6330, 1820, 419900, 2775, 23423, single family
marry a. massano, (858)943-2249, 1973, 475000, 2340, 23551, single family
deborah reddington, (508)882-7166, 1954, 599000, 2321, 20413, single family
justin ryan rollo, (617)274-8931, 1950, 379900, 1104, 21184, single family

6 append the code of the Python script to the end of this pdf file

```
import csv
import math
A_mat = []
B_mat = []
TABLE_A_DIR="table_A2.csv"
TABLE_B_DIR="table_B2.csv"
#define leagal year range
BEGIN_YEAR=1800
END_YEAR=2017

def load_table_A(table_A_dir):
  #load data from table A
  with open(table_A_dir, 'rb') as A:
    spamreader1 = csv.reader(A, delimiter=',', quotechar='|')
    for row in spamreader1:
      A_mat.append(row)
  return A_mat

def load_table_B(table_B_dir):
  #load data from table B
  with open(table_B_dir, 'rb') as B:
      spamreader2 = csv.reader(B, delimiter=',', quotechar='|')
      for row in spamreader2:
        B_mat.append(row)
  return B_mat

def data_merging_and_creating_file(A_mat, B_mat):
  #implement data merging between data matrix A and B
  #write the merged and combined table into new matrix E
  with open("table_E.csv", 'wb') as csvOut:
    writer = csv.writer(csvOut,delimiter = ',', quotechar='|')
    A_mat = A_mat[1:]
    B_mat = B_mat[1:]
    l = len(A_mat)
    if l != len(B_mat):
      print "error, not same length"
    for i in range(l):
      row = []
      row1 = A_mat[i]
      row2 = B_mat[i]
      #split name into first, middle, last name
      f1,m1,l1 = "","",""
```

```
      f2,m2,l2 = "","",""
      name = ""
      name1 = row1[0]
      name2 = row2[0]
      phone1 = row1[1]
      phone2 = row2[1]
      year1 = row1[2]
      year2 = row2[2]
      #begin merging between attributes of names
      name1 = name1.split()
      name2 = name2.split()
      #implement name merging rule
      f1 = name1[0].lower()
      if len(name1) == 3:
        m1 = name1[1].lower()
      l1 = name1[-1].lower()

      f2 = name2[0].lower()
      if len(name2) == 3:
        m2 = name2[1].lower()
      l2 = name2[-1].lower()

      if len(f1) > len(f2):
        name += f1
      else:
        name += f2
      name += " "
      if len(m1) > len(m2):
        name += m1
        name += " "
      else:
        name += m2
        name += " "

      if len(l1) > len(l2):
        name += l1
      else:
        name += l2

      row.append(name)

      # matching phone number
      real_phone1 = ""
      for c in phone1:
        if c.isdigit():
          real_phone1 += c

      real_phone2 = ""
      for c in phone2:
        if c.isdigit():
          real_phone2 += c
      phoneNum = ""
      if len(real_phone1) == 10:
        phoneNum = "(" + real_phone1[:3] + ")" +real_phone1[3:6] + "-" +
    real_phone1[6:]
      elif len(real_phone2) == 10:
        phoneNum = "(" + real_phone2[:3] + ")" +real_phone2[3:6] + "-" +
    real_phone2[6:]

      row.append(phoneNum)
      year_int_1 = 100000
      year_int_2 = 100000
      #merging attribute of year
      year1 = year1.strip()
      year2 = year2.strip()
      if year1.isdigit():
        year_int_1 = int(year1)
      if year2.isdigit():
        year_int_2 = int(year2)

      if year_int_1 in range(BEGIN_YEAR, END_YEAR) and year_int_2 not in range
    (BEGIN_YEAR, END_YEAR):
        year = year_int_1
```

```
113         elif year_int_1 not in range(BEGIN_YEAR, END_YEAR) and year_int_2 in
        range(BEGIN_YEAR, END_YEAR):
114           year = year_int_2
115         else:
116           year = min(year_int_2, year_int_1)
117         if year == 100000:
118           year = ""
119
120         row.append(year)
121
122         row.append(row1[3])
123         row.append(row1[4])
124         row.append(row2[3])
125         row.append(row2[4])
126         #write the merged value into data table E
127         writer.writerow(row)
128
129 if __name__ == "__main__":
130   A_mat = load_table_A(TABLE_A_DIR)
131   B_mat = load_table_B(TABLE_B_DIR)
132   data_merging_and_creating_file(A_mat, B_mat)
```

Listing 1: Python Script for Combination