CS 839 Stage 4 Report
Data Analysis on the Integrated Table

Xiuyuan He, Chenlai Shi, Mingren Shen

1. Creating Table E

First, we run the blocker and matcher we got from stage 3 to determine the matches between table A and table B. After this step, we get a matched set which contains ***1337*** matches.

Before merging two tables, we carefully look through the matched set and we found that even for the matched tuples, the information may not be the same. However, due to the simplicity of our data, for each field, we can just keep the longer data, since it usually contains much more information. For example, for a matched book in table A and table B (*Data Science from Scratch: First Principles with Python*), one of the record only contains the main title (*Data Science from Scratch*, this is because the webpage splits title into two lines to display a book with main title in one line and sub-title in another line, however our web scraper only records the first line) whereas another one contains the whole long title. In such case, we believe that keep longer title more suitable.

The second problem is that the 1337 matches are not strictly one-to-one match, e.g. one book in table A may match 3 books in table B, so here we have to decide how to deal with such cases. We follow the merging rules above and we believe this contains the majority of the information. And then when merging the rest of the unmatched tuples from table A and table B, we only keep the tuples that did not show in the matched set, because we believe our merging rule has already recorded information in those tuples and we only need to keep the unique unmatched remaining ones.

In terms of the schema of the new merging Table E, we keep 7 fields from table A and table B, and we add 1 new field called Source which denotes the source of this tuple: ‘a’ if the tuple is from table A, ‘b’ is it’s from table B, and ‘ab’ if it’s from the the matched tuples in A and B:

1) Title: book title, string => choose the longer one
2) Author: book authors, string => choose the longer one
3) Publication:information of publishers, string => choose the longer one
4) Format: book types, category, e.g. journal, magazines => choose the longer one

5) ISBN: Search Results International Standard Book Number,integer => choose the longer one
6) Series: book series information, string => choose the longer one
7) Physical Details:Physical description in book cataloging, string => choose the longer one
8) Source : Use `a` denotes the tuple from table A, `b` denotes the tuple from table B and `ab` denotes the tuple is contained in both table A and table B.

So after merging all the matches, we add all the remaining tuples in table A and table B that did not show in the matched set. So after completing populating the table E, we get the final table E which has ***9350*** tuples.

Examples of Table_E are shown below.

```
set_merged
```

	Title	Author	Publication	Format	ISBN	Series	Physical Details	Source
0	"statistical learning and data science"	"edited by mireille gettler summa ... [and others]"	"boca raton : crc press, [2012] ©2012"	"books"	9781439867631	"series in computer science and data analysis,"	"xv, 227 p."	ab
1	"intelligent techniques for data science"	"rajendra akerkar, priti srinivas sajja"	"cham, switzerland : springer, 2016."	"electronic books."	9783319292069	"nan"	"1 online resource (xvi, 272 pages)"	ab
2	"algorithms for data science"	"brian steele, john chandler, swarna reddy"	"cham, switzerland : springer, 2016."	"electronic books."	9783319457970	"nan"	"1 online resource (xxiii, 430 pages)"	ab
3	"data science at the command line"	"janssens, jeroen"	"first edition. sebastopol, ca : o'reilly, 2014. ©2015"	"books"	9781491947852	"nan"	"xvii, 191 pages"	ab

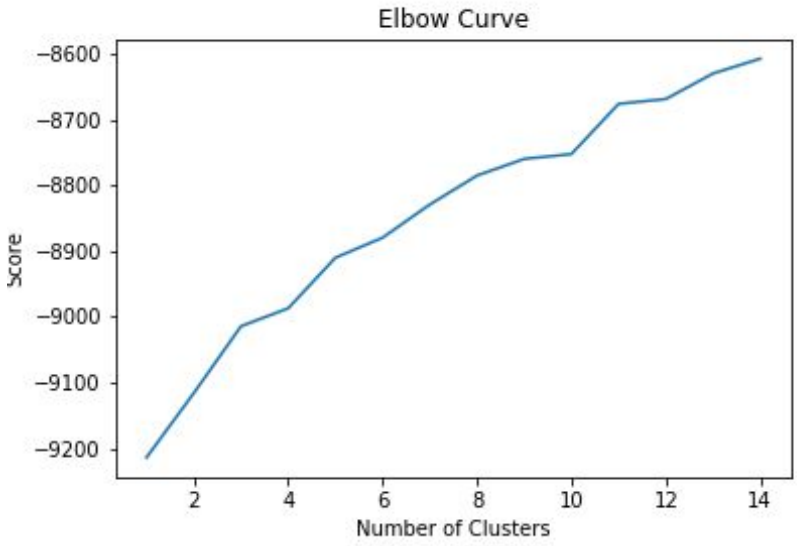
2. Data Analysis

We want to analyse the book repositories of UIUC and UW-Madison.Because we do not have the category information of the books, so we want to use K-means clustering to categorized the books and get some insights of the collections from different libraries. For example, which category of books UIUC likes more than UW-Madison and which category is liked by both universities.

We measure the term frequencies with TF-IDF based on the book titles to get the feature vector of each book and then we use k-means method in scikit-learn to cluster each book in Table_E into 10 clusters.

For k-means clustering, the most important hyper-parameter is k, which is the number of clusters to be used in clustering. We use Elbow method to choose appropriate k for the clustering. The Elbow method is a method of interpretation and validation of consistency within cluster analysis designed to help finding the appropriate number of clusters in a dataset. In one word, Elbow method tries to find the point where at some point the marginal gain will drop, giving an angle in the graph.

However, the graph does not show a very clear shape angle in the crvue. And this is because This "elbow" cannot always be unambiguously identified. Taking the complexity of the clustering and the explanation of clusters into consideration, here we choose 10 as the value of k because it can best explain the meaning of each cluster.



To explain the meaning of each cluster, we extract the top terms per cluster and summarized below, we can clear see 2 major categories one is the book(Cluster 0,1,2,3,5,7,8), another one is the collection of conference papers(Cluster 4,6,9).

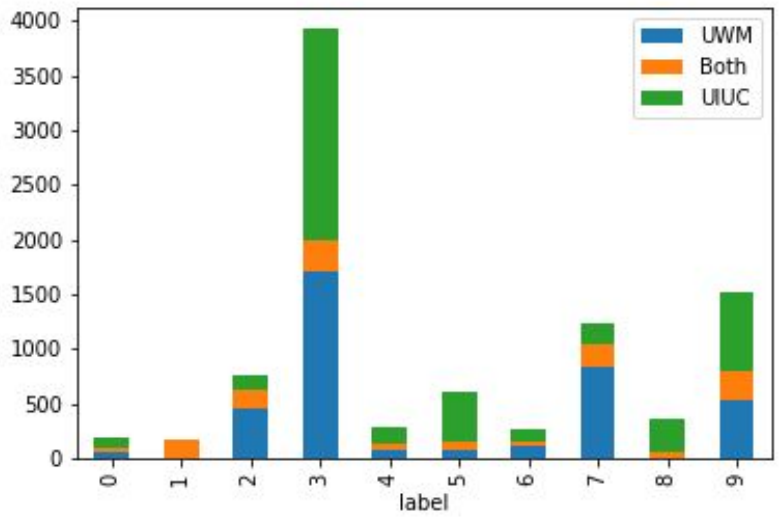
Cluster 0	Handbook, data, science, research, sage, analysis, methods, social, oxford, library
Cluster 1	Science, computer, data, computational, engineering, transactions, applications, social, research
Cluster 2	Structures, data, algorithms, java, using, introduction, pascal, algorithm, analysis, programming
Cluster 3	Data,sciences,applications,systems,information,research,methods,computing,social,introduction
Cluster 4	2012,international,conference,proceedings,september,revised,selected,papers,information,october
Cluster 5	Digital,multimedia,hiding,security,age,transactions,forensics,data,language,information

Cluster 6	Proceedings,international,conference,2013,2017,2015,2011,september,information,cryptology
Cluster 7	Centered,scale,large,transactions,knowledge,systems,data,issue,special,expert
Cluster 8	Analysis,data,social,methods,guide,sciences,spss,statistical,using,statistics
Cluster 9	Papers,selected,revised,2016,international,2014,conference,2011,workshop,security

The bar plot below shows all the statistics of Table_E. For cluster 0, 4, 6, 9 where the tuples are likely to be conference papers, we can see that the fraction of book from different libraries are similar. Furthermore, for cluster 2 and 7, where the topic is related to algorithms, systems and programming languages, the fractions of book from UWM are significantly higher. Similarly, for cluster 5 and 8, the fractions of book from UIUC are significantly higher. Finally, we summarized several most favorable topics for each library based on the clusters.

For UWM, the most common book topics are in cluster 2, 3, 7 and 9, where the frequent seen terms include: data structures, algorithms, data analysis, large scale, conference paper, applications and so on.

For UIUC, the common topics are in cluster 3, 5, 8 and 9, where the frequent seen terms include: data sciences, systems, applications, security, data analysis, conference paper and so on.



If we have more time, we will combine data from outside sources like book reviews from Amazon or Google Books to further identity the book qualities of each university. Also, if we can cooperate with libraries of each university, we can combine borrowing history data from their

database, we can analyse the reading habit of each university and recommend books to different users. Because we know their reading habit and know the other users, so both Collaborative filtering and Content-based filtering can be used here.

Appendix

Python script used to merge the tables:

```
# coding: utf-8
# # Stage 4 Data Analysis on the Integrated Table

# First, we repeat what we did in stage 3 and get a set of matching tuples.
import pandas as pd
import py_entitymatching as em
A = em.read_csv_metadata('./TableA_lower.csv', key = 'ID')
B = em.read_csv_metadata('./TableB_lower.csv', key = 'ID')

block_f = em.get_features_for_blocking(A, B)
block_t = em.get_tokenizers_for_blocking()
block_s = em.get_sim_funs_for_blocking()
r = em.get_feature_fn('jaccard(dlm_dc0(ltuple["Title"]),
dlm_dc0(rtuple["Title"]))', block_t, block_s)
em.add_feature(block_f, 'Title_Title_jac_dlm_dc0_dlm_dc0', r)

ob = em.OverlapBlocker()
C = ob.block_tables(A, B, 'Author', 'Author',

l_output_attrs=['Title','Author','Publication','Format','ISBN','Series',
'Physical Details'],

r_output_attrs=['Title','Author','Publication','Format','ISBN','Series',
'Physical Details'],
                    overlap_size = 2)

D = ob.block_candset(C, 'Title', 'Title', overlap_size = 4)
label_S = pd.read_csv('./Set_G.csv')
# em.copy_properties(S, label_S)
em.set_property(label_S, 'key', '_id')
em.set_property(label_S, 'fk_ltable', 'ltable_ID')
em.set_property(label_S, 'fk_rtable', 'rtable_ID')
```

```
label_S_rtable = em.read_csv_metadata('./label_S_rtable.csv')
label_S_ltable = em.read_csv_metadata('./label_S_ltable.csv')
em.set_property(label_S, 'rtable', label_S_rtable)
em.set_property(label_S, 'ltable', label_S_ltable)
label_S['ltable_Title'] = label_S['ltable_Title'].apply(lambda x:
x.lower())
label_S['rtable_Title'] = label_S['rtable_Title'].apply(lambda x:
x.lower())
label_S['ltable_Author'] = label_S['ltable_Author'].apply(lambda x:
x.lower())
label_S['rtable_Author'] = label_S['rtable_Author'].apply(lambda x:
x.lower())
label_S['ltable_Publication'] = label_S['ltable_Publication'].apply(lambda
x: x.lower())
label_S['rtable_Publication'] = label_S['rtable_Publication'].apply(lambda
x: x.lower())
label_S['ltable_Format'] = label_S['ltable_Format'].apply(lambda x:
x.lower())
label_S['rtable_Format'] = label_S['rtable_Format'].apply(lambda x:
x.lower())
label_S['ltable_Series'] = label_S['ltable_Series'].apply(lambda x:
x.lower())
label_S['rtable_Series'] = label_S['rtable_Series'].apply(lambda x:
x.lower())
match_f = em.get_features_for_matching(A, B)
match_t = em.get_tokenizers_for_matching()
match_s = em.get_sim_funs_for_matching()
f1 = em.get_feature_fn('jaccard(dlm_dc0(ltuple["Title"]),
dlm_dc0(rtuple["Title"]))', match_t, match_s)
# f2 = em.get_feature_fn('jaccard(dlm_dc0(ltuple["Author"]),
dlm_dc0(rtuple["Author"]))', match_t, match_s)
f3 = em.get_feature_fn('jaccard(dlm_dc0(ltuple["Publication"]),
dlm_dc0(rtuple["Publication"]))', match_t, match_s)
f4 = em.get_feature_fn('jaccard(dlm_dc0(ltuple["Series"]),
dlm_dc0(rtuple["Series"]))', match_t, match_s)
em.add_feature(match_f, 'Title_Title_jac_dlm_dc0_dlm_dc0', f1)
# em.add_feature(match_f, 'Author_Author_jac_dlm_dc0_dlm_dc0', f2)
em.add_feature(match_f, 'Publication_Publication_jac_dlm_dc0_dlm_dc0', f3)
em.add_feature(match_f, 'Series_Series_jac_dlm_dc0_dlm_dc0', f4)

# Add black box feature
import re
```

```
# for Roman numerals matching
def Title_Title_blackbox_1(x, y):
    # get name attribute
    x_title = x['Title']
    y_title = y['Title']
    regex_roman = '\s+[mdclxvi]+($|\s+)'
    x_match = None
    y_match = None
    if re.search(regex_roman, x_title):
        x_match = re.search(regex_roman, x_title).group(0).strip()
    if re.search(regex_roman, y_title):
        y_match = re.search(regex_roman, y_title).group(0).strip()
    if x_match is None or y_match is None:
        return False
    else:
        return x_match == y_match

em.add_blackbox_feature(match_f, 'blackbox_1', Title_Title_blackbox_1)

# for number matching (e.g. 6th edition)
def Title_Title_blackbox_2(x, y):
    # x, y will be of type pandas series
    x_title = x['Title']
    y_title = y['Title']
    regex_number = '\s+(\d+)\s*th'
    x_match = None
    y_match = None
    if re.search(regex_number, x_title):
        x_match = re.search(regex_number, x_title).group(1)
    if re.search(regex_number, y_title):
        y_match = re.search(regex_number, y_title).group(1)
    if x_match is None or y_match is None:
        return False
    else:
        return x_match == y_match

em.add_blackbox_feature(match_f, 'blackbox_2', Title_Title_blackbox_2)

# for number matching (e.g. 6th edition)
from fuzzywuzzy import fuzz
def Author_Author_blackbox_3(x, y):
    # x, y will be of type pandas series
```

```
    x_author = x['Author']
    y_author = y['Author']
    return fuzz.token_set_ratio(x_author, y_author)/100.0

em.add_blackbox_feature(match_f, 'blackbox_3', Author_Author_blackbox_3)
match_f = match_f[(match_f['left_attribute'] != 'ID') &
(match_f['left_attribute'] != 'ISBN')]
match_f = match_f[(match_f['left_attribute'] != 'Format') &
(match_f['left_attribute'] != 'Series')]
H = em.extract_feature_vecs(label_S, feature_table=match_f,
attrs_after=['label'])
# RF
rf = em.RFMatcher(n_estimators = 300,max_depth = 300, name='RF')
rf.fit(table=H,
       exclude_attrs=['_id', 'ltable_ID', 'rtable_ID', 'label'],
       target_attr='label')
H_test = em.extract_feature_vecs(D, feature_table=match_f)
pred_table = rf.predict(table= H_test,
                        exclude_attrs=['_id', 'ltable_ID', 'rtable_ID'],
                        target_attr='predicted_labels',
                        return_probs=True,
                        probs_attr='proba',
                        append=True)
# eval_summary
pred_rows = pred_table[pred_table['predicted_labels'] == 1]

# ## Merge the tables
#
# Here we obtained the matching set based on the prediction.
#
#
# There are 1337 matched books in total.
matched_set = D[D['_id'].isin(pred_rows['_id'])]
matched_set
matched_set.shape
matched_id = matched_set[['ltable_ID', 'rtable_ID']]

# We put the id of matching tuples in to dictionaries.

AB_list = matched_id.values.tolist()
AB_dict =  {item[0]: item[1] for item in AB_list}
BA_dict =  {item[1]: item[0] for item in AB_list}
```

```
len(AB_dict.keys())

# Here we merge the matching tuples.
#
# For each field of a matching tuple, we simply keep the one with longer
string length.

df = matched_set.iloc[:, 3:10]
df = df.rename(columns={'ltable_Title' : 'Title',
                  'ltable_Author' : 'Author',
                   'ltable_Publication' : 'Publication',
                   'ltable_Format' : 'Format',
                   'ltable_ISBN' : 'ISBN',
                   'ltable_Series' : 'Series',
                   'ltable_Physical Details' : 'Physical Details'
                  })

attr=['Title','Author','Publication','Format','ISBN','Series', 'Physical
Details']
i = 0
for index, r in matched_set.iterrows():
    for  a in attr:
        left_name = 'ltable_' + a
        right_name = 'rtable_' + a
        if len(str(r[left_name])) < len(str(r[right_name])):

            df.iloc[i, attr.index(a)] = r[right_name]
        else:
            df.iloc[i, attr.index(a)] = r[left_name]
    i += 1

# We added an Source attribute to indicate the source of a tuple.

df['Source'] = 'ab'
A['Source'] = 'a'
B['Source'] = 'b'

# After merging the matching set, we concat the set with the original table
A and B.

set_merged = pd.concat([df, A[~A['ID'].isin(AB_dict.keys())].iloc[:,1:],
B[~B['ID'].isin(BA_dict.keys())].iloc[:,1:]])
```

```
A[~A['ID'].isin(AB_dict.keys())].shape
B[~B['ID'].isin(BA_dict.keys())].shape
set_merged

# Final merged set contains 9350 tuples in total.

# The final set contains 9350 tuples in total.
set_merged.shape
set_merged[set_merged['Source'] == 'a'].shape

# We save the set to Table_E.csv.
set_merged.to_csv('Table_E.csv', sep=',')
```