# CS 839 Stage 4

Lan Bai, Chaoqun Mei, Yuzhe Ma

May 8, 2018

**Abstract** In this project stage, we merged the two tables extracted from Amazon and Barnes & Noble and then did analysis. We clustered the entities in the merged table into 5 clusters using K-means clustering and Partitioning Around Medoids (PAM), which is less sensitive to outliers compared to K-means.

## 1. Merge of A and B

To merge A and B, we first apply the matcher we obtained in Stage 3 to all pairs after blocking to drop the unmatched pairs, so we get a table with all matched pairs, which contains the ID of matched tuples in A and B. Then we scan A, check the ID of the tuple, if the ID does not exist in the table of matched pairs, we append that tuple to the end to B.

We do not have other tables.

ID: Number ID of book in the table, from 1 to N (number of tuples).

Publisher: The book's publisher.

Time: The book's publish date.

Author: The book's author's name.

Title: The book's title (name).

## 2. Statistics of E

We use an AttrEquivalenceBlocker and an OverlapBlocker. We assume that the same books should have same publish date, so we use AttrEquivalenceBlocker to block tuple pairs whose Time attributes are not the same (We let the Time to be the same format in preprocessing). We also assume that the same books should have overlap of at least one word in their title, so we use OverlapBlocker to block tuples whose Title attributes have less than 1-word overlap.

Schema of E is the same as schema of A and B, which has 5 attributes:

ID: Number ID of book in the table, from 1 to N (number of tuples).

Publisher: The book's publisher.

Time: The book's publish date.

Author: The book's author's name.

Title: The book's title (name).

There are 6317 tuples in total.

Example:

6524,Touchstone,1/2/2018,Rob Barnett,The Economists' Diet: The Surprising Formula for Losing Weight and Keeping It Off

6525,Wiley; 1 edition,4/14/2014,Than Merrill,"The Real Estate Wholesaling Bible: The Fastest, Easiest Way to Get Started in Real Estate Investing"

6528,Harper Wave,3/13/2018,Leah Weiss PhD,"How We Work: Live Your Purpose, Reclaim Your Sanity, and Embrace the Daily Grind"

6529,Keys to the Vault; First edition,12/15/2017,Keith J. Cunningham,The Road Less Stupid

6530,Zondervan,4/25/2017,Bill High,Giving It All Away__nd Getting It All Back Again: The Way of Living Generously

## 3. Feature Construction for E

To perform learning-based analysis on the books crawled from online sources, we first construct feature representation for each book, i.e. each row in the combined table E. The way we construct features is based on the word-to-vector representation of the vocabulary formed by the titles of all the books. To illustrate the principle in detail, let us pretend E only has two rows, where the first row is the book titled "Mathematical Analysis" and the second row is the book titled "English Learning". We construct a vocabulary for the above two titles in two steps.

Step 1: do text preprocessing to turn all the letters in the titles into lowercase, followed by removing extra white spaces. Thus in the above case, we turn the original two titles into "mathematical analysis" and "english learning".

Step 2: identify all distinct words that appear in any book titles, and combine them together into a set of distinct words, which is called a vocabulary. In the above case, we end up with a four-item vocabulary {mathematical, analysis, english, learning}. Note that we do not perform any word stemming in our case, i.e. "learning" is not turned into "learn".

After we have the vocabulary V, we use the popular word-to-vector method to assign an M-dimensional vector representation for each word in V. Refer to the following paper for details.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

In our experiment, M=50. Now we have an M-dimensional representation $\alpha_i$ for word $i$. Then in order to create a representation for book $b$, we first take its title $T_b$, which we treat as a word set. Then the representation is computed as follows.

$$X_b = \frac{1}{|T_b|} \sum_{i \in T_b} \alpha_i,$$

which is basically the average of the vectors for those words that appear in the title. We apply the above process to the titles of all N books and generate an N by M matrix, where each row is the representation for each book.

## 4. Data Analysis on E

### 4.1 K-means clustering on E

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data.

2. Labels for the training data (each data point is assigned to a single cluster).

In the clustering of E, we choose K = 5 based on our extraction of the books from Amazon and Barnes & Noble, the maximum iteration is 100, and Euclidean distance is used. The final cluster of each data point is listed in the X_cluster.csv file in the Data folder. We use within-cluster sum-of-squares and between-cluster sum-of-squares to evaluate the final cluster, within-cluster sum-of-squares are listed in Table 1, between-cluster sum-of-squares is 20166132354. A good clustering yields clusters which have small within-cluster sum-of-squares and high between-cluster sum-of-squares. We can see that the between-cluster sum-of-squares is much larger than the total within-cluster sum-of-squares, which indicates that the final cluster is pretty good.

Table 1 within-cluster sum-of-squares for K-means clustering

| Cluster | 1 | 2 | 3 | 4 | 5 | tot.withinss |
|---|---|---|---|---|---|---|
| withinss | 168294301 | 167495720 | 167895632 | 167897037 | 168693575 | 840276265 |

### 4.2 Partitioning Around Medoids (PAM) clustering on E

Partitioning Around Medoids algorithm is intended to find a sequence of objects called medoids that are centrally located in clusters. Objects that are tentatively defined as medoids are placed into a set S of selected objects. If O is the set of objects that the set U = O − S is the set of unselected objects. The goal of the algorithm is to minimize the average dissimilarity of objects to their closest selected object. Equivalently, we can minimize the sum of the dissimilarities between object and their closest selected object.

The algorithm has two phases:

(i) In the first phase, BUILD, a collection of k objects is selected for an initial set S.

(ii) In the second phase, SWAP, one tries to improve the quality of the clustering by exchanging selected objects with unselected objects.

In the Partitioning Around Medoids clustering of E, we also set the number of clusters at 5, and the Euclidean distance as the metric. The final cluster of each data point is also listed in the X_cluster.csv file in the Data folder. The final cluster information of PAM is listed in Table 2.

size: the cardinality of the cluster (number of observations).

max_diss: the maximal dissimilarity between the observations in the cluster and the cluster's medoid.

av_diss: the average dissimilarity between the observations in the cluster and the cluster's medoid.

diameter: the diameter of the cluster (maximal dissimilarity between two observations of the cluster).

separation: the separation of the cluster (minimal dissimilarity between an observation of the cluster and an observation of another cluster).

Table 2 Final cluster information of PAM

| size | max_diss | av_diss | diameter | separation |
|---|---|---|---|---|
| 1262 | 631.0022 | 315.5225 | 1261.001 | 1.803383 |
| 1264 | 632.0017 | 316.0176 | 1263.001 | 1.803383 |
| 1264 | 632.0051 | 316.0294 | 1263.002 | 2.427889 |
| 1263 | 632.002 | 315.7725 | 1262.003 | 2.795528 |
| 1264 | 633.0028 | 316.0167 | 1263.002 | 2.87269 |

From the separation in Table 2, we can see that the final cluster is pretty good.

**Appendix**

Code of Merge

```
Feature = em.extract_feature_vecs(C,
                              feature_table=F,
                              show_progress=False);
# In[23]:



Feature = em.impute_table(Feature,
                exclude_attrs=['_id', 'ltable_ID', 'rtable_ID'],
                strategy='mean');



# In[24]:



pred = Y.predict(table=Feature, exclude_attrs=['_id', 'ltable_ID', 'rtable_ID'],
              append=True, target_attr='predicted', inplace=False);



# In[34]:



Matched = [0]*(len(A) + 1)
drop_unmatch = []
for index, row in pred.iterrows():
    if row['predicted'] == 1:
        Matched[int(row['ltable_ID'])] = 1;
    else:
        drop_unmatch.append(index)
all_match = C.drop(drop_unmatch)
em.to_csv_metadata(all_match, './Matches.csv')



# In[27]:



Result = B
id = 3862
for index, row in A.iterrows():
    if Matched[row['ID']] == 0:
        row['ID'] = id
        id = id + 1
        Result.loc[row['ID']-1]=row
em.to_csv_metadata(Result, './E.csv')
```