CS 839 – Project stage 4 Report

Team Members:

- Sreyas Krishna Natarajan(snatarajan6@wisc.edu)
- Sugadev Chellakkannu(chellakkannu@wisc.edu)
- Vignesh Thirunavukkarasu(thirunavukka@wisc.edu)

Show how did you combine the two tables A and B to obtain E? Did you add any other table? When you did the combination, did you run into any issues? Discuss the combination process in detail, e.g., when you merge tuples, what are the merging functions?

We performed the merging process differently for each set of attributes from the matched table.

We carefully examined each attribute of both the tables and merged them in a way which will best reflect their meaningfulness. We now describe the helper functions that we used to merge the attributes:

Merge Function:

merge_tuple(tupleA, tupleB):

The function receives tuples from Table A and Table B. It checks for the names of each Column; Title, Artist, Genre, Release, Rating and does the following process.

The Merging Process for Each attribute is follows:

Title:

The attribute value which had the longest length was picked inclusive of the values inside the brackets. That's because the values in the brackets add more descriptive value to this field.

Eg*: How Big, How Blue, How Beautiful [Bonus Tracks] [CD]*, *reputation [Picture Disc] [2 LP] [LP] – VINYL*, *Thank You [Deluxe Edition] [CD]*

Artist:

The attribute value which had the longest length of both tables was picked.

We merged it in this way because:

- The Artist field may sometimes be missed of co-author names.
- It may have a short version of author name instead of the author's full name.

Genre:

In BestBuy collections this field is populated with one-word value whereas Metacritic had more specific in terms of their categories. So, we decided to merge the values from two tables. Thus, it provides more detailed roots of the album.

Eg:

BestBuy	Blurryface [CD]	Pop
Metacritic	Blurryface	Pop, Rap, Pop/Rock, Alternative/Indie Rock, Alternative Pop/Rock, Indie Rock, Underground Rap

Release Date:

The values in both the tables represent the same date but in different format.

Eg: 5/15/2001, 15-May-01 → We chose the former. If the first field is empty, then we format the date from the other table and populate.

Rating:

BestBuy : Rating is out of 5. The rating is partly based on the quality of album and the quality of CD's, DVD's also

Metacritic : Rating is out of 10. The rating is based solely on viewers review on the quality of the album.

So, we chose the attribute value of Metacritic where the rating is based solely on the album.

Issues

- In the Column field "Rating", some of the values are marked *TBD (To be decided).* Then, we populated those attribute values to 0.
- Empty date from one table was populated with the dates from the other table.
- When choosing Rating on a scale of 5 or 10, we were initially confused as to which one we should choose. We then used the notion of how ratings are made on the corresponding websites to choose.
- We did not add any other new column during the merging stage. Once the merging was complete and we had a final single table E. The code for the merging process can be found at the end of this document.

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.

Schema

S No.	Attribute	Type	Description
1	Title	String	Title of the album
2	Artist	String	Name of the artist
3	Genre	String	Song Category
4	Release	String	Date published. Format – month/day/year
5	Rating	Int	Rating out of 10

Number of tuples

Table E has 101 tuples.

Sample:

Title	Artist	Genre	Release	Rating
reputation [CD]	Taylor Swift	Pop, Adult Contemporary, Pop/Rock, Dance-Pop, Teen Pop	11/10/2017	7.7
Camila [CD]	Camila Cabello	Pop, Pop/Rock, Dance-Pop, Pop Idol, Teen Pop	1/12/2018	8.9
Unapologetically [CD]	Kelsea Ballerini	Pop, Country, Pop/Rock, Contemporary Country, Country-Pop	11/3/2017	7.5
The Neighbourhood [CD]	The Neighbourhood	Pop, Electronic, Pop/Rock, Contemporary Pop/Rock, Left-Field Pop, Synthwave	3/9/2018	0.0

What was the data analysis task that you wanted to do? For that task, describe in detail the data analysis process that you went through.

Data Analysis:

We wanted to find patterns of change in the average rating of different sub-genres (Contemporary, Alternative, Rock) of Pop over the years of 2000-2017

Description:

1) Grouping of the tuples by the year from Release date.
2) Extracting the respective sub-genres (Contemporary, Alternative, Rock, Pop) from the Genre
3) Find the average rating for a specific sub-genre and the number of albums with that sub-genre released
4) Print these average rating and number of releases over the years 2000-2017.

Give any accuracy numbers that you have obtained (such as precision and recall for your classification scheme). Report which learning based matcher you selected after that cross validation.

We did not perform a classification scheme. We performed OLAP analysis and discussed the results in the next section.

What did you learn/conclude from your data analysis? Were there any problems with the analysis process and with the data?

We performed OLAP analysis using pandas dataframe as discussed in the previous sections. The results are as below

Data:

2000

	Genre	count	mean
0	Pop	1	8.5

2001

	Genre	count	mean
0	Alternative	1	8.80
1	Contemporary	2	7.65
2	Pop	1	7.50

2002

	Genre	count	mean
0	Alternative	1	8.9
1	Contemporary	1	7.8
2	Pop	2	8.1
3	Rock	1	8.2

2003

	Genre	count	mean
0	Pop	1	8.1

2004

	Genre	count	mean
0	Contemporary	1	7.8

2005

	Genre	count	mean
0	Alternative	1	9.0
1	Contemporary	1	6.2
2	Pop	1	6.0

2006

	Genre	count	mean
0	Contemporary	1	7.40
1	Pop	2	7.35

2007

	Genre	count	mean
0	Alternative	1	8.1
1	Contemporary	2	7.8
2	Rock	1	7.8

2008

	Genre	count	mean
0	Alternative	2	8.6500
1	Rock	8	8.1875

2009

	Genre	count	mean
0	Pop	1	7.700000
1	Rock	3	8.266667

2010

	Genre	count	mean
0	Alternative	1	9.0
1	Pop	2	5.4

2011

	Genre	count	mean
0	Alternative	1	7.9
1	Contemporary	1	6.7
2	Rock	1	6.2

2012

	Genre	count	mean
0	Contemporary	8	7.7375
1	Pop	1	7.3000
2	Rock	2	6.7000

2013

	Genre	count	mean
0	Contemporary	1	7.800000
1	Rock	3	8.533333

2014

	Genre	count	mean
0	Contemporary	4	7.35
1	Pop	2	8.60
2	Rock	3	8.60

2015

	Genre	count	mean
0	Alternative	3	8.366667
1	Contemporary	4	7.650000
2	Rock	5	7.000000

2016

	Genre	count	mean
0	Alternative	2	7.600000
1	Contemporary	11	6.936364
2	Pop	1	8.400000
3	Rock	3	8.466667

2017

	Genre	count	mean
0	Alternative	2	8.200000
1	Contemporary	12	7.741667
2	Pop	2	7.150000
3	Rock	4	7.550000

2018

	Genre	count	mean
0	Contemporary	1	0.0
1	Rock	1	8.9

Findings:

- We found that the average Rating of Alternative Pop/Rock genres had reduced through the years. The sub-genre used to average a Rating of 8.5+ during the years 2000-2011. It went down to 8.0+ after 201 4. This might be due to emergence of numerous albums in other sub-genres.

- We couldn't see a specific pattern with Rock as it has had its ups and downs. Still remain one of the mo st popular genres.

- Contemporary music always averages a rating from 6.5 – 7.9

If you have more time, what would you propose you can do next?

On careful analysis of the dataset, we found that there are multiple names that refers to the same Genre.

Eg. Pop and Generic Pop, country and country pop. They refer to the **same real-world entity**. And some are represented using the **synonyms**. We could build a dictionary that represents variations of the same genre names or train to classifier to do this task.

Append the code of the Python script (that merges the tables) to the end of this pdf file.

```
#Initializing The matchers

dt = em.DTMatcher(name='DecisionTree', random_state=0)

svm = em.SVMMatcher(name='SVM', random_state=0)

rf = em.RFMatcher(name='RF', random_state=0)

lg = em.LogRegMatcher(name='LogReg', random_state=0)

ln = em.LinRegMatcher(name='LinReg')

nb = em.NBMatcher(name='NaiveBayes')


# The features for matching

F = em.get_features_for_matching(A, B, validate_inferred_attr_types=False)


# List the names of the features generated

F['feature_name']


#Extract Feature Vectors.

H = em.extract_feature_vecs(I,
```

```
                            feature_table=F,
                            attrs_after='Match',
                            show_progress=False)

# compare stats to select best Matcher
#Logistic Regression in our case

result = em.select_matcher([dt, rf, svm, ln, lg, nb], table=H,
        exclude_attrs=['_id', 'ltable_ID', 'rtable_ID', 'Match'],
        k=5,
        target_attr='Match', metric_to_select_matcher='f1', random_state=0)
result['cv_stats']



#Read in the complete candidate set obtained after blocking (973 tuples)
C = em.read_csv_metadata("/mnt/c/Users/sreya/Downloads/candidates_large1.csv",
                         key='_id',
                         ltable=A, rtable=B,
                         fk_ltable='ltable_ID', fk_rtable='rtable_ID')
C
K = em.extract_feature_vecs(C, feature_table=F,show_progress=False)
# Impute feature vectors with the mean of the column values
K = em.impute_table(K, exclude_attrs=['_id', 'ltable_ID', 'rtable_ID'],
                 strategy='mean')

# fit with best matcher which was logistic regression
lg.fit(table=H, exclude_attrs=['_id', 'ltable_ID', 'rtable_ID', 'Match'],target_attr='Match')

# take best classifier (logistic regression) and output predictions (i.e. matches)
predictions = lg.predict(table=K, exclude_attrs=['_id', 'ltable_ID', 'rtable_ID'],
                         append=True, target_attr='predicted', inplace=False)

# save set of matches between two tables
matches = predictions.loc[predictions['predicted'] == 1]
```

```
matches = matches.loc[:,'_id'].tolist()
mask = C['_id'].isin(matches)
mat = C.loc[mask]
mat.drop_duplicates()
#matches table
mat
#mat.to_csv("matches.csv")

#Splitting the indices of ltable(Metacritic) and rtable(BestBuy)
metacriticMatchInd = [int(s)-1 for s in list(predictions.loc[predictions['predicted'] == 1,'ltable_ID'])]
BBMatchInd = [int(s)-1 for s in list(predictions.loc[predictions['predicted'] == 1,'rtable_ID'])]

#Creating an empty dataframe with columns as in A
E = pd.DataFrame(columns = A.columns[1:])

#merging rtable and ltable values to get E
def mergeTuples(x1, x2):
    t = []
    for i in np.arange(1,len(x1)):
        if A.columns[i] in ['Title','Artist','Genre']:
            if len(x1[i]) >= len(x2[i]):
                curr = x1[i]
                t.append(curr)
            else:
                curr = x2[i]
                t.append(curr)
        if A.columns[i] in ['Release']:
            t.append(x2[i])
        if A.columns[i] in ['Rating']:
            if(x1[i] == 'tbd'):
                t.append(0.0)
            else:
                t.append(float(x1[i]))
    return t
```

```
#Calling mergetuples on tuples from BestBuy and Metacritic
for i in np.arange(0,len(BBMatchInd)):
    BBTuple = B.loc[BBMatchInd[i]][0:]
    #print(BBTuple)
    metaTuple = A.loc[metacriticMatchInd[i]][0:]
    #print(metaTuple)
    merged = mergeTuples(BBTuple, metaTuple)
    E.loc[i] = merged

#Merged Table = E
E.to_csv('E.csv')
```