CS838 PROJECT REPORT - STAGE 4

COMBINING TWO TABLES

(04/ 16/ 2017)

Gaurav Mishra	Om Jadhav	Pallavi M Kakunje
gmishra2@wisc.edu	ojadhav@wisc.edu	kakunje@wisc.edu

1. Overview:

The goal of this project stage is to combine the matching tuples i.e., tuples that represent same entity, from two tables into a single table. In this stage, we used the same data sources as in the previous stages.

Entity type: Mobile game application

Data Sources: Google Play Store (google table) and Apple Store (apple table).

We use the matches obtained from Stage 3 of the project and combine the corresponding matched tuples in this stage.

2. Implementation:

2.1. Combining Process:

Our best matcher as identified from the previous stage (Stage 3) is Random Forest. This matcher is applied to all the candidate matches that survived the blocking stage. This provided all the matches. Resulting matches are stored in all_predictions.csv.

Our python script goes through all the predictions to find whether the tuples are predicted as match or not. If there is a match, then we copy corresponding tuples from both the tables into matched_tuples.csv. Further we merge both tuples according to merger rules and copy the merged tuple into merged_tuples.csv (table E).

This stage involved combining just the two above mentioned tables and no additional table.
The process of combining was smooth, without facing any unexpected issues.

2.2 Merging rules and Schema

Both google table and apple table have the same schema of the form:
(“ID”, “name”, “category”,” developer”, “rating”)
The final merged table also has the same schema. The attribute values are merged using following rules: ID takes new values in the merged table E. “name” and ”developer” are same in both table if there is a match. So, we copy it from one of the table (google table is the chosen one). After examining the tables, we observed that “category” in google table makes more sense as compared to “category” in apple table. So we pick category from google table. “Rating” is computed as the average value of “rating” values from both the tables. There are few tuples in which rating is not present in apple table, in this case we pick rating from google table alone.

2.3. Merged Table (Table E):

Both google table and apple table have same schema. So, merged_tuples table (table E) will also have same schema as mentioned above. Each tuple of E contains attributes: “ID”, “name”, “category”,” developer” and “rating”.

There are total 490 tuples in table E.

Few tuples of table E are as follows:

ID	name	category	developer	rating
1	Candy Crush Saga	Casual	King	4.554035
2	Super Mario Run	Action	Nintendo Co., Ltd.	3.33734
3	Jetpack Joyride	Arcade	Halfbrick Studios	4.421695
4	Temple Run	Arcade	Imangi Studios	4.170195
5	Subway Surfers	Arcade	Kiloo	4.496675
6	Hill Climb Racing	Racing	Fingetsoft	4.423215
7	Sonic Dash	Arcade	SEGA	4.49338
8	Super Stickman Golf 2	Sports	Noodlecake Studios Inc	4.41765

3. Python script:

merger.py

```
import re

#all_predictions.csv is contains predicted output for all potential matches.
#In this script, we are checking for each potential match whether it is
#predicted as match or not. It it is matched then we are copying both tuples
#in matched_tuples.csv. We are merging both tuples according to merger rules
#and copying the merged tuple in merged_tuples.csv.
fq = open('all_predictions.csv','r')
fp = open('merged_tuples.csv','w')
ft = open('matched_tuples.csv','w')

#first line in merged_tuples.csv file for providing headings.
a = '"ID",'+'"name",'+'"category",'+'"developer",'+'"rating"'
fp.write(a)
fp.write('\n')

#first line in matched_tuples.csv file for providing headings.
#Each line in this file contains matched tuples from both google table and apple table.
a = '"G_ID",'+'"G_name",'+'"G_category",'+'"G_developer",'+'"G_rating",'+\
    '"A_ID",'+'"A_name",'+'"A_category",'+'"A_developer",'+'"A_rating"'
ft.write(a)
ft.write('\n')

#initializing j here which will act as ID.
j = 0

for line in fq:
    word = line.split()
    #checking if the rows are predicted as matched or not.
    if(word[len(word)-1] == "1"):
        #writing ID in the first field
        j = j + 1
        idx = '"'+str(j)+'",'
        fp.write(idx)

        #extracting row number of each tuple which are being matched.
        word_google = word[2]
        word_apple = word[3]

        #variables to hold attribute values of google table
        ID_google = ""
        name_google = ""
        category_google = ""
        developer_google = ""
        rating_google = ""

        #variables to hold attribute values of apple table
        ID_apple = ""
        name_apple = ""
        category_apple = ""
        developer_apple = ""
        rating_apple = ""

        with open('google.csv') as fr:
            with open('apple.csv') as fs:
                #based on the extracted row number, reading the corresponding row
                #from google table and writing whole row in matched_tuples.csv file.
                content_google = fr.read().splitlines()
                row = content_google[int(word_google[1:-1])]
                ft.write(row)
                ft.write(",")

                #extracting different attribute values for merging.
                #extracting ID.
                start = row.find('"') + 1
                end = row.find('"', start)
                ID_google = row[start:end]

```

```
                #extracting name.
                row = row[(end+1):]
                start = row.find('"') + 1
                end = row.find('"', start)
                name_google = row[start:end]

                #extracting category.
                row = row[(end+1):]
                start = row.find('"') + 1
                end = row.find('"', start)
                category_google = row[start:end]

                #extracting developer.
                row = row[(end+1):]
                start = row.find('"') + 1
                end = row.find('"', start)
                developer_google = row[start:end]

                #extracting rating.
                row = row[(end+1):]
                start = row.find('"') + 1
                end = row.find('"', start)
                rating_google = row[start:end]

                #based on the extracted row number, reading the corresponding row
                #from apple table and writing whole row in matched_tuples.csv file.
                content_apple = fs.read().splitlines()
                row = content_apple[int(word_apple[1:-1])]
                ft.write(row)
                ft.write('\n')

                #extracting different attribute values for merging.
                #extracting ID.
                start = row.find('"') + 1
                end = row.find('"', start)
                ID_apple = row[start:end]

                #extracting name.
                row = row[(end+1):]
                start = row.find('"') + 1
                end = row.find('"', start)
                name_apple = row[start:end]

            #extracting category.
            row = row[(end+1):]
            start = row.find('"') + 1
            end = row.find('"', start)
            category_apple = row[start:end]

            #extracting developer.
            row = row[(end+1):]
            start = row.find('"') + 1
            end = row.find('"', start)
            developer_apple = row[start:end]

            #extracting rating.
            row = row[(end+1):]
            start = row.find('"') + 1
            end = row.find('"', start)
            rating_apple = row[start:end]

            #if both google table and apple table contain rating then taking
            #average of both ratings. Otherwise, taking rating from google table.
            if((rating_apple.find('Ratings') == -1) and (rating_apple.find('none') == -1)):
                rating_google = (float(rating_google) + float(rating_apple))/2

            #combining all attributes as a single string and writing it in merged_tuples.csv.
            to_write = '"'+name_google+'",'+'"'+category_google+'",'+\
                       '"'+developer_google+'",'+'"'+str(rating_google)+'"'
            fp.write(to_write)
            fp.write('\n')
    fp.close()
    fq.close()
    ft.close()
```