839 Project Stage 4: Matching Movie Entity Integration and Analysis

Daniel K. Griffin
Department of Computer Science
University of Wisconsin Madison
dgriffin5@wisc.edu

Yudhister Satija
Department of Computer Science
University of Wisconsin Madison
ysatija@wisc.edu

Mitali Rawat
Department of Computer Science
University of Wisconsin Madison
mitali.rawat@wisc.edu

1 Introduction

In this project stage we performed data integration across our data sources, building on the entity matcher that was developed in the previous stage between two data tables holding movie information. The sources that this data was collected from is provided below, and is the same as the data sources for project stage 2. Below, we provide a quicklist that can be used for quickly grading our assignment. We provide some further details in later sections.

2 Quick List

1. **The Web data sources**: We used the following 2 sources:
   - IMDB movies list from http://www.imdb.com/list/ls032600534
   - Movie Numbers list from https://www.the-numbers.com/movies/#tab=letter
2. **How did we combine the tables A,B to form E?:** Once we have chosen the blocker and matcher (in stage 3), we applied this to the tables A and B and obtained a table of the matcher outputs. This table, called M, now contains the pair of movies that the matcher predicted same. The schema of this table is (ltable_id, ltable_title, rtable_id, rtable_id) and contains 2674 tuples. We use this table to construct the integrated table E.
3. **Did we add another table?:** No
4. **What where the issues we ran into?:** Most of the issues we ran into dealt with how to clean data, and how to merge data. Issues such as normalizing movie ratings and categories, how gross numbers are represented, and how attributes should be merged into a single table.
5. **Discuss the Integration Process:** We first wrote functions that would return the merged value for a feature from a pair of matched movies. Details:
   (a) **Movie title:** We pick whichever of the two movie titles is longer (no missing values in this column)
   (b) **Year:** We pick the year value from table B, unless if it's missing in which case we pick from table A.
   (c) **MPAA rating:** We pick this rating from table A, unless if its missing in which case we use value from table B.
   (d) **Runtime:** We pick the larger runtime value of the two.
   (e) **Directors:** We do a union over all director names from both tables.

(f) **Stars:** We do a union over all stars names from both tables.
(g) **Gross:** We first convert all gross values to numbers by converting suffixes like 'M' and removing the '$' symbol. Then we pick the larger value of the two.
(h) **Genre:** We do a union over all genres mentioned for both movies.

With these functions at hand, we went through the ltable_id and rtable_id values picked from table M. Using each id, we picked the tuple from the corresponding table. We pass the pair of tuples to the above functions and obtain the integrated single tuple. We do this for all tuples listed in M and save to make table E.

6. **Number of Tuples in E?:** 2674
7. **4 Samples from E:**

	id	title	year	mpaa	runtime	genres	director	stars	gross
0	1757	Hamlet	1996	PG-13	242 min	drama	Kenneth Branagh	julie christie,derek jacobi,kate winslet,kenne...	$441000000.0
1	3547	Home for the Holidays	1995	PG-13	104 min	drama,comedy, romance	jodie foster	charles durning,robert downey jr.,holly hunter...	$1752000000.0
2	502	Hot Fuzz	2007	R	121 min	action,comedy, comedy	edgar wright	simon pegg,martin freeman,bill nighy,nick frost	$2364000000.0
3	3688	Drop Dead Gorgeous	1999	PG-13	97 min	comedy, romance, thriller	Michael Patrick Jann	kirsten dunst,ellen barkin,denise richards,all...	$1056000000.0
4	3517	Bye Bye Birdie	1963	Approved	112 min	musical, musical,comedy	George Sidney	Dick Van Dyke,Ann-Margret,Janet Leigh,Maureen ...	$1313000000.0

Figure 1: Sample from integrated data

8. **What is the schema of table E?:** Table Columns: id, title, year, mpaa, runtime, genres, director, stars, gross.
9. **Describe our Analysis Process:** Our analysis process consisted of performing aggregated, categorical OLAP analysis in relation to movie gross and movie runtime. One can imagine the scenario of being a data scientist at a movie production studio. Some of the business goals at such a company might be to determine a portfolio of movies to make for a year that would maximize expected gross for the year, and minimize risk. Thus, some interesting questions might be, "What kinds of movie genres make the most money", and "What kinds of movie rating categories have movies that make the most money". Our OLAP and correlation analysis follows these kinds of questions. It would also be relatively simple to extend this analysis into an optimization or machine learning problem for building an actual portfolio.
10. **Give any accuracy numbers that you have obtained:** We performed correlation analysis and OLAP style analysis. So, the numbers we present in the item below are related to aggregated metrics rather than performance metrics. We provide plots for our OLAP style analysis, and correlation analysis numbers.
11. **What did we learn/conclude from your data analysis?:**
    (a) **Runtime correlation to Movie Gross:** SpearmanrResult(correlation=0.25852659016321544, pvalue=4.341047669255873e-42). Our analysis shows that the movie runtime is fairly correlated to the movie gross, and that the alternative hypothesis that they are not correlated is very unlikely.
    (b) **Year correlation to Movie Gross:** SpearmanrResult(correlation=0.3388931633734646, pvalue=7.458510545431806e-73). Our analysis shows that the movie year is fairly correlated to the movie gross, and that the alternative hypothesis that they are not correlated is very unlikely.
    (c) **Movie gross per genre kind:** Figure 1 below shows a set of box plots for movie gross in Billions based on movie genre kinds. A box plot provides multiple pieces of interesting information (which is why I like using them). A box plot shows median, average, and quantiles for the data which provides insight about data centrality, distribution spread, and outliers. The plot shows interesting information, such as adventure and animation have a higher amount of movies in the upper quantiles than any other movie genre kind. Action movies also seem to have an average near that of adventure and anime, but has a higher spread (which biases the average). This might indicate that while action movies on average make less money, there is more of a change that they will be blockbusters. Other interesting information is that more niche movies like film-noir tend to make less money.

(d) **Movie times per genre kind:** Figure 2 below shows a set of box plots for movie times based on movie genre. History movies seem to be the longest on average. Dramas also seem to have a lot of outlier movies that are longer than other genres, even though the average is about the same as other movies. We can also see that the "short" movie genre also has a lower median and average runtime, with a low spread, which helps to validate what we would expect in our data set.

(e) **Movie gross per mpaa:** Figure 3 below shows a set of box plots for movie gross in billions based on movie mpaa rating. The plot shows that pg and pg-13 movies tend to make more money both on average, and in terms of outliers (blockbusters). This might indicate why most movie studios try to edit their movies to either get the pg or pg-13 movie rating. It actually might also indicate why the movie rating "pg-13" was initially introduced since the movie ratings were originally only "pg" and "r".

(f) **Movie times per mpaa:** Figure 4 below shows a set of box plots for movie runtime based on movie mpaa rating. This plot seems to show that there isn't a discernible relationship between movie runtime and movie rating.

3 Analysis Plots

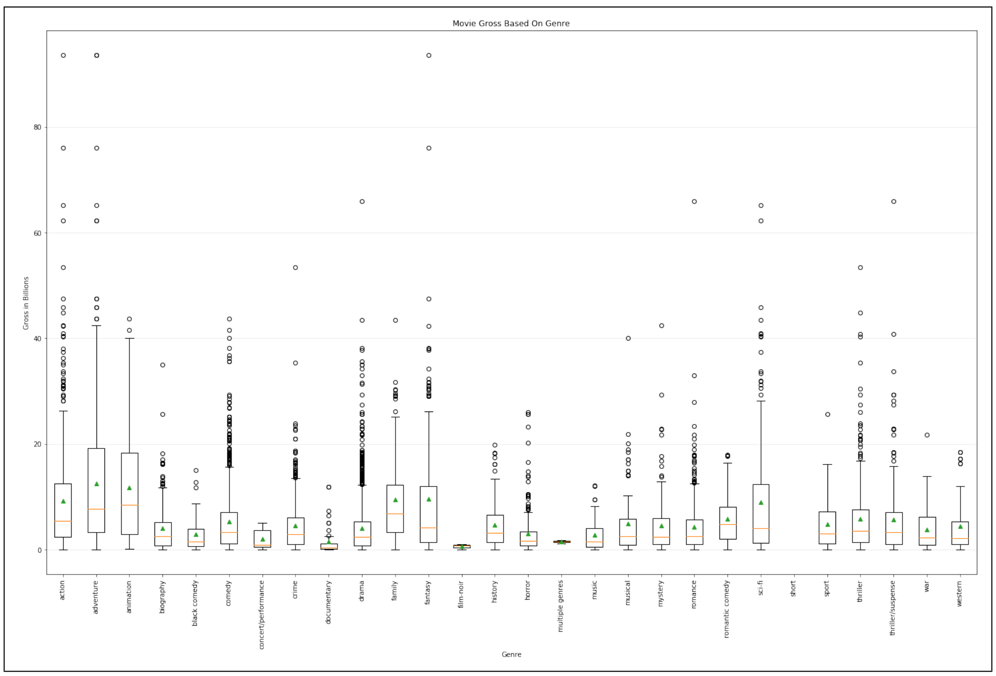

Figure 2: Movie gross per genre kind

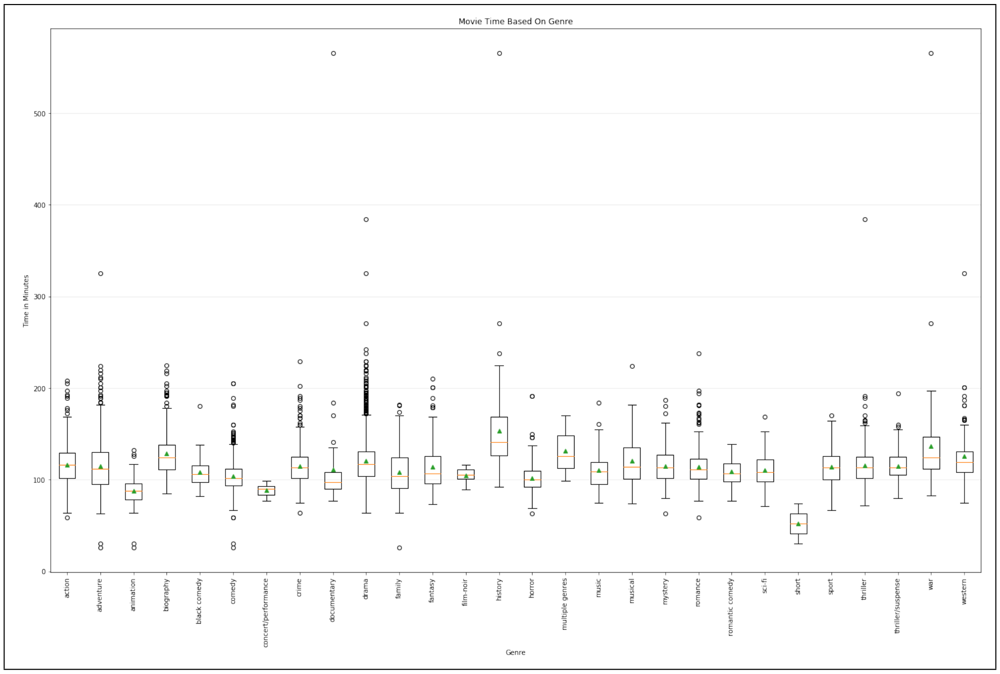

Figure 3: Movie times per genre kind

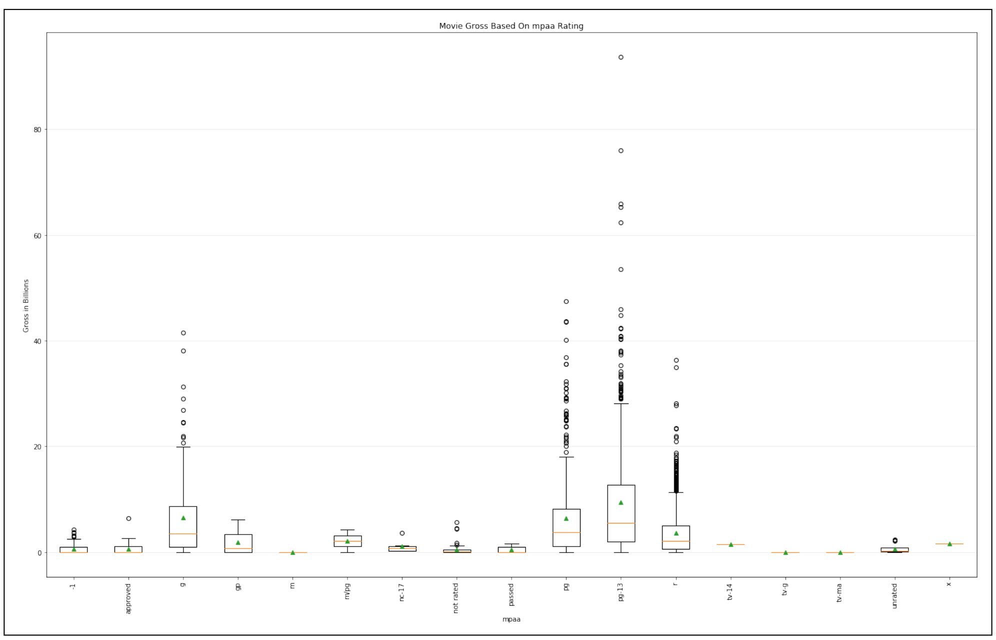

Figure 4: Movie gross per mpaa

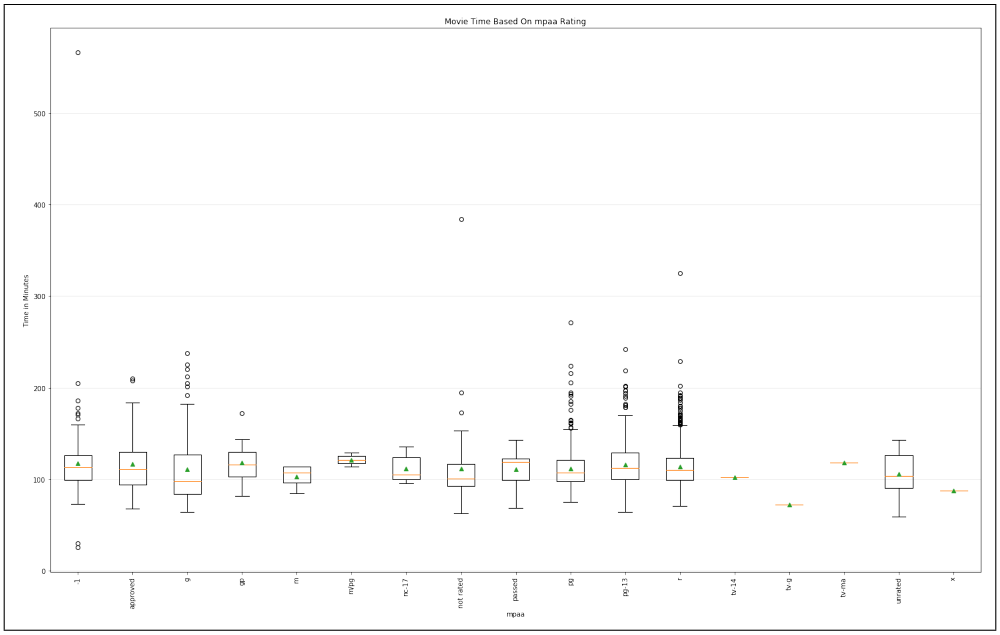

Figure 5: Movie times per mpaa

4 Python Integration Scripts

4.1 "merge_data.py"

This script is run to generate the integrated_table.csv using table A, B and M

```
import sys
import string
import re
import math
import pandas as pd

path_A = '../DATA/imdb3_neg_nan.csv'
path_B = '../DATA/thenumbers3_neg_nan.csv'
path_M = '../DATA/MatchPredctionsOnAllTuplePairs.csv'
A = pd.read_csv(path_A)
B = pd.read_csv(path_B)
Matches = pd.read_csv(path_M)
print(len(Matches))
#Matches.head()

def merge_title(title1, title2):
    #take the longer title
    title1= title1.item()
    title2=title2.item()
    l1 = len(title1)
    l2= len(title2)
    res = title1 if l1>l2 else title2
    return res

def merge_year(year1, year2):
    #if value form B exists, take it else from A
    if year2.item() != -1:
        return year2.item()
```

```
    return year1.item()

def merge_mpaa(mpaa1,mpaa2):
    mpaa1=mpaa1.item()
    mpaa2=mpaa2.item()
    return mpaa1 if mpaa1 != "Not Rated" and mpaa1 != "-1" else mpaa2

def merge_runtime(rt1,rt2):
    rt1=rt1.item()
    rt2=rt2.item()
    regex = re.compile('[^0-9]')
    a=regex.sub('',rt1)
    b=regex.sub('',rt2)
    rt1 = int(a)
    rt2 = int(b)
    rt = rt1 if rt1>rt2 else rt2
    return str(rt)+" min"

def split_and_union(g1,g2):
    g1 = g1.lower()
    g2= g2.lower()
    g1 = g1.split(",")
    g2= g2.split(",")
    final_list = g1+g2
    final_list = set(final_list)

    return ','.join(final_list)

def merge_genres(g1,g2):
    g1=g1.item()
    g2=g2.item()
    if g1=="-1": return g2
    if g2=="-1": return g1
    return split_and_union(g1,g2)

def merge_director_name(dir1,dir2):
    dir1=dir1.item()
    dir2=dir2.item()
    if dir1=="-1": return dir2
    if dir2=="-1": return dir1
    return split_and_union(dir1,dir2)

def merge_stars(stars1, stars2):
    stars1=stars1.item()
    stars2=stars2.item()
    if stars1=="-1": return stars2
    if stars2=="-1": return stars1
    return split_and_union(stars1,stars2)

def merge_gross(grossL, grossR):
    grossL=grossL.item()
    grossR=grossR.item()
    #remove the '$' or any other special character from gross value of right table-
    #is of the form "$1234,123"
    if grossR == "-1" and grossL == "-1":
        return grossL
    grossRint=grossLint=0
    if grossR != "-1":
        grossclean2 = ''.join(ch for ch in grossR if ch in string.digits)
```

```
            grossRint = int(grossclean2)
        #grossL is of the form "$4.4M"
        if grossL != "-1":
            dictL={'M':1e6,'B':1e9,'T':1e12,'k':1e3, 'K':1e3}
            grossL.replace(" ","")
            if grossL[-1] in dictL:
                    multfact=dictL[grossL[-1]]
            grossclean1 = ''.join(ch for ch in grossL if ch in string.digits)
            grossLint = float(grossclean1)
            grossLint *= multfact
        f = grossLint if grossLint>grossRint else grossRint
        return "$"+str(f)

# Combine two movie tuples
def combine(lt, rt):

    a = merge_title(lt.title, rt.title)
    b = merge_year(lt.year, rt.year)
    c=merge_mpaa(lt.mpaa, rt.mpaa)
    d=merge_runtime(lt.runtime, rt.runtime)
    e=merge_genres(lt.genres, rt.genres)
    f=merge_director_name(lt.director, rt.director)
    g=merge_stars(lt.stars, rt.stars)
    h=merge_gross(lt.gross, rt.gross)
    return (lt.id.item(),a,b,c,d,e,f,g,h)

# pick each tuple from M and take out corresponding
# tuples from A and B, and call combine on them.
finalist=[]
for row in Matches.itertuples():
    #print(row)
    lid = row.ltable_id
    rid = row.rtable_id
    ltup = A.loc[(A["id"]==lid)]
    rtup = B.loc[(B["id"]==rid)]
    tup = combine(ltup, rtup)
    # Append to the final table
    finalist.append(tup)
df = pd.DataFrame(finalist, columns=['id', 'title', 'year', 'mpaa','runtime','genres
# Save to file
df.to_csv('../DATA/integrated_table.csv', index=False)
```