Magic Movie Predictors and Sentiment Exploration

Peng Wu, Guanshun Yu
Department of Biostatistics, Department of Electrical Engineering
Columbia University
pw2394@cumc.columbia.edu, gy2226@columbia.edu

Abstract—**This project primarily aims at selecting import predictive features to predict movie ratings and clustering various movies for future recommendation system based on users' preference. Sentiment analysis and keyword analysis on good movies is our secondary goal.**

Keywords: Movie Ratings, Random Forests, SVM, K-Means, DBSCAN, Sentiment Analysis, Word Cloud

I. INTRODUCTION

Nowadays, more and more people are choosing to buy tickets for movies. We are immersed in a world of all kinds of movies and we all like better experience of movie watching. Audiences are keep trying to find a more reliable way to determine the quality of a movie. On the other hand, a movie producer may also want to know how will the movie perform when it goes on market. Indeed, using machine learning techniques along with sentiment analysis can help to explore this.

II. RELATED WORKS

The first information a movie customer will look into before going to a theater is movie rating and its corresponding reviews. How accurate is a movie rating is? What factors are most likely going to affect a movie rating? How we want to cluster movies based on users' preference? What are the reviews like about a specific movie? Are they more positive or negative? These are all questions we are interested in this project. Related works ([2], [3], [5], [6], [7]) are done to find a way to answer these and in our work we want to explore more.

III. SYSTEM OVERVIEW

We have three main data sources. The first one is contained in R package "ggplot2movie" which includes large amount of movie information until 2005. This dataset is a well-formatted subset from plain text files provided by IMDB database. The entire dataset has 58788 observations and more than twenty variables (e.g. ratings, production year, movie genres etc.). The second one is by calling IMDB unofficial API to get the reviews. The third one is collected by using twitter for illustration purpose.

We further cleaned this dataset by using SQL-like language in R. The data is parsed by some subset criteria, for instance, votes over 2000. The features we are interested in are built correspondingly. With the help of proximity from random forest, the missing variables are imputed. The final dataset contains over 3000 observations with more than 10 predictive features.

Preliminary visualization plots are displayed in Figure 1 and 2. The movie ratings are our primary outcome of interests. The ratings are normally distributed so it is robust to use regression techniques for data analysis. The movie budgets will be a very import features in our case and they are almost exponentially distributed. The bubble plot in Figure 2 is quite interesting. Most Rate-R movies are coming after 1990 and the number of movies are increasing significantly by years. Correspondingly, more recent movies have a larger budget which can be directly seen from this plot. An interesting phenomenon here is that ratings varies more and more with increasing time. That means we have a more spread-out type of movies after 2000, a mixed world of good and bad movies.

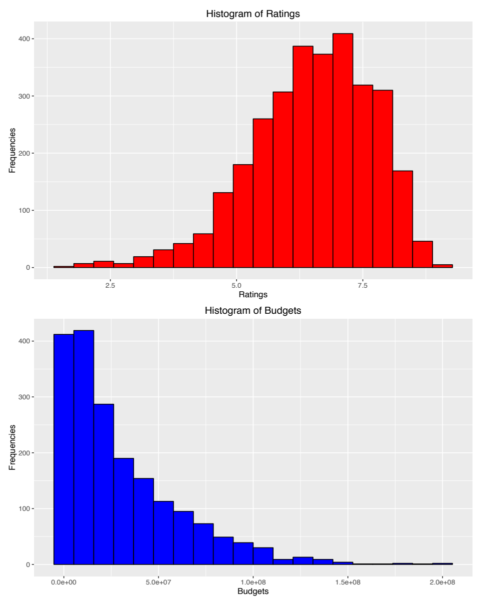


Figure 1. Preliminary Visualization (Histogram)

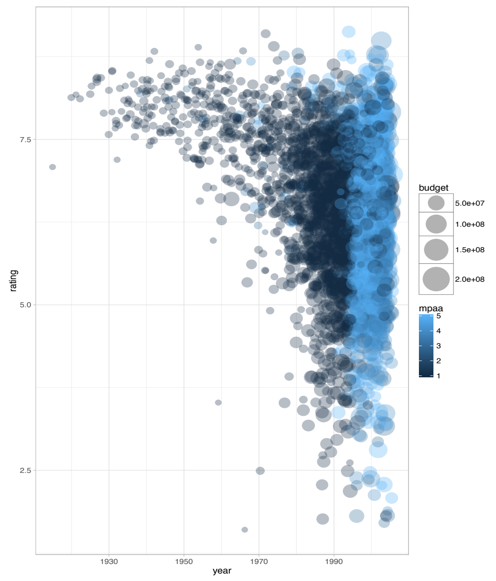


Figure 2. Preliminary Visualization (Scatterplot)

IV. ALGORITHM

Statistical Learning

In the statistical learning part, four learning algorithms are performed in the first dataset. Two are supervised learning with ratings as the label: random forests and support vector machine (SVM). The other two are unsupervised clustering methods by removing the rating labels, which are K-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The clustering can be view as an unsupervised version of classification.

For regressions, the data is split into training set and testing set by random sampling. The testing set size is about ½ of training one. In random forest regression, the number of trees choosing is based on the error rate in order to avoid overfitting problem. In Figure 3, 80 should be sufficient. Similarly, SVM is conducted and the tuning parameters are find by using cross-validation.

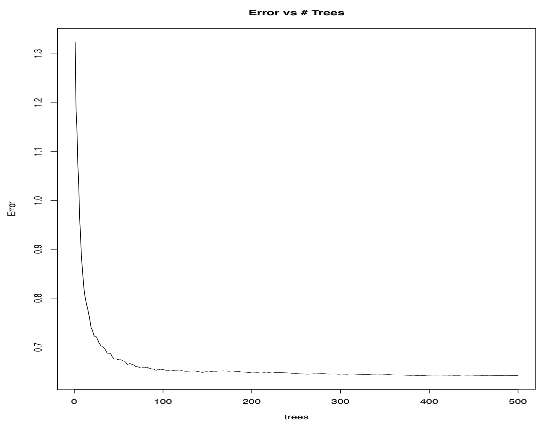


Figure 3. Random Forests Tree Number Choice

For clustering, we use the entire cleaned dataset by deleting true ratings. In K-means, the optimal K is selected by looking into the "kink" of total within cluster sum of squares. In DBSCAN, we utilized weighted version by computing weight as the fraction of votes and production year subtracted from 2010. The intuition here is that the most recent and most voted movies will get higher attention in clustering. There are some noise points existed after clustering and we keep the clusters with over 200 points.

Sentiment Analysis

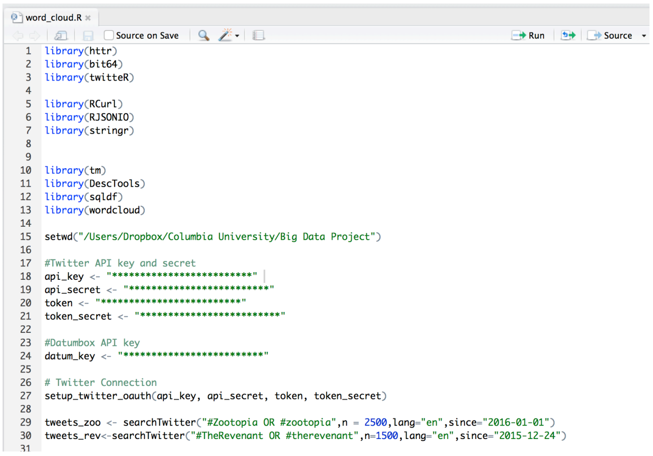
After collecting data of top 250 movies, we extract 15 reviews from each movie. Next, Alchemy API is used to performed sentiment analysis on all the reviews. Then we took the average score of all 15 reviews and assigned it to the movie. Moreover, we put top 250 movies into different categories according to their genres variable attribute. After categorizing top 250 movies, we not only look at the amount of movies in each category but also calculated the sentiment score for each category.

For illustration purpose, we collect over 2000 tweets from twitter about a two well-known movies since last year, "The Zootopia" and "The Revenant". One is a highly recommended animation in 2016 and the other is an Academy big winner.

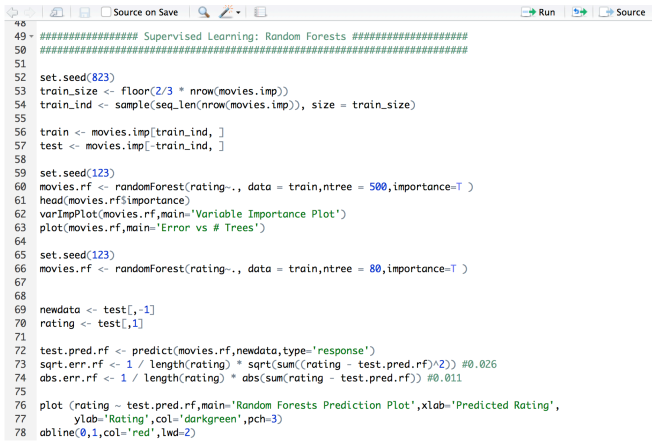
V. SOFTWARE PACKAGE USE DESCRIPTION

Statistical learning methods are performed in R (randomForest, e1071, dbscan etc.) and sentiment analysis and exploration are done in Python (imdbpie). Tools such as Alchemy API and Datumbox API are utilized.

For instance, an example of how to use "twitteR" to stream twitter information about "The Zootopia" and use R package "randomForests" to predict movie ratings are as below:



```r
library(httr)
library(bit64)
library(twitteR)

library(RCurl)
library(RJSONIO)
library(stringr)


library(tm)
library(DescTools)
library(sqldf)
library(wordcloud)

setwd("/Users/Dropbox/Columbia University/Big Data Project")

#Twitter API key and secret
api_key <- "************************"
api_secret <- "************************"
token <- "************************"
token_secret <- "************************"

#Datumbox API key
datum_key <- "************************"

# Twitter Connection
setup_twitter_oauth(api_key, api_secret, token, token_secret)

tweets_zoo <- searchTwitter("#Zootopia OR #zootopia",n = 2500,lang="en",since="2016-01-01")
tweets_rev<-searchTwitter("#TheRevenant OR #therevenant",n=1500,lang="en",since="2015-12-24")
```



```r
################# Supervised Learning: Random Forests ###################
#########################################################################

set.seed(823)
train_size <- floor(2/3 * nrow(movies.imp))
train_ind <- sample(seq_len(nrow(movies.imp)), size = train_size)

train <- movies.imp[train_ind, ]
test <- movies.imp[-train_ind, ]

set.seed(123)
movies.rf <- randomForest(rating~., data = train,ntree = 500,importance=T )
head(movies.rf$importance)
varImpPlot(movies.rf,main='Variable Importance Plot')
plot(movies.rf,main='Error vs # Trees')

set.seed(123)
movies.rf <- randomForest(rating~., data = train,ntree = 80,importance=T )


newdata <- test[,-1]
rating <- test[,1]

test.pred.rf <- predict(movies.rf,newdata,type='response')
sqrt.err.rf <- 1 / length(rating) * sqrt(sum((rating - test.pred.rf)^2)) #0.026
abs.err.rf <- 1 / length(rating) * abs(sum(rating - test.pred.rf)) #0.011

plot (rating ~ test.pred.rf,main='Random Forests Prediction Plot',xlab='Predicted Rating',
      ylab='Rating',col='darkgreen',pch=3)
abline(0,1,col='red',lwd=2)
```

Below is a part of sentiment score calculation.

```python
from imdbpie import Imdb
imdb = Imdb()
imdb = Imdb(anonymize=True) # to proxy requests

import pickle
import ast
import json
movie=open('movie review score_0-50','r')
movie1=open('movie review score_51-100','r')
movie2=open('movie review score_101-150','r')
movie3=open('movie review score_151-200','r')
movie4=open('movie review score_201-250','r')
#dictionary=dict(x.split(':') for x in movie.read().split('],'))
#print dictionary
Scores= eval(movie.read())
Scores.update(eval(movie1.read()))
Scores.update(eval(movie2.read()))
Scores.update(eval(movie3.read()))
Scores.update(eval(movie4.read()))
print len(Scores)
#temp=[]
#temp.append(float(Scores['tt1675434'][0]))
#print temp
MovieSentScore={}
id=[]

#average=[]
for i in Scores:
    temp=[]
    MovieSentScore[i]=[]

    #id.append(i)
    for j in range(len(Scores[i])):
        temp.append(float(Scores[i][j]))
    #average.append(sum(temp)/len(Scores[i]))
    MovieSentScore[i].append(sum(temp)/len(Scores[i]))
    #print len(average)
    #rint len(id)
```

VI. Experiment Results

Statistical Learning

For random forests regression, we get top 5 importance variables, these variables are the most important in building the trees (as splitting criteria): movie votes, production year, movie budget, movie length and indicator of action movies. The regressor on testing set does a relatively good job with L1 error equals 0.011 and L2 error equals to 0.026. For example, on training set, the predicted rating for "*Star Wars*" is 8.2 and the true rating is 8.8; the predicted rating for "*American Beauty*" is 8.0 and the true rating is 8.5; the predicted one for "*Schindler's List*" is 8.3 with true one equals to 8.8. We note that there is an underestimate effect which can be explained by the "regression-to-mean" trend. Thus for prediction, we can add some noise terms to boost the variance. Also it is worth to note that the prediction is more accurate among movies with higher ratings, which inspire us to look into the sentiment analysis for top-rated movies, as no one loves bad movies.

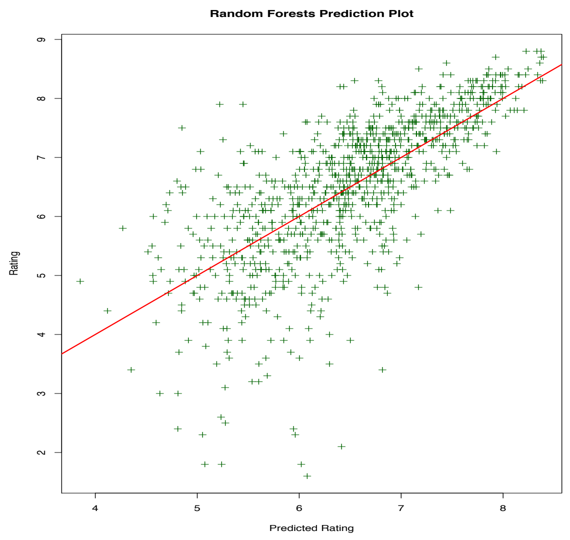


Figure 4. Random Forests Prediction Accuracy

For SVM, the tuning parameters are cost =1 and gamma = 0.0625 (visualized in Figure 5). Similar prediction accuracy procedures are gone through. The error rate control is good too with L1 error = 0.06 and L2 error = 0.026. Thus we the regression is reasonable to explain the relationship between our outcome and features. Hence these predictive features well explain the association between movie ratings and themselves.

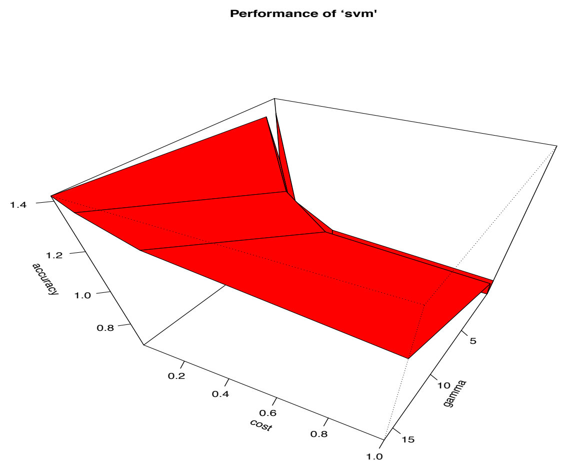


Figure 5. SVM Tuning Parameters Choice

In unsupervised learning, we remove movie rating from the features set. For K-means clustering, first we select the optimal K by computing the within cluster sum of squares. From Figure 6 about the sum of squares curve, the first two curve bends appear at K=2 or K=4. So we choose K=4 in our case. Other methods like gap statistics proposed by Tibishirani can also be used to select optimal K. Figure 7 displays a visualization of clustering in two main dimensions which explain over 32% data variability. The cluster results are shown in Table 1 from which we can find some useful information. The "good" Academy nominated movies tend to be cluster together with higher average rating, the examples include "*Pulp Fiction*" and "*The Godfather*". The animations are clustered together as well with a better rating 7.0, examples are "*Shrek*" and "*Finding Nemo*". Another cluster contains most likely commercial movies such as "*The Matrix*" and "*The Lord of the Rings*", these movies may have a relatively larger budgets. The rest potentially less well-known movies are clustered in a group.

Clusters	Average Ratings	Examples
Cluster 1	7.1	The Shawshank Redemption, Pulp Fiction, The Godfather, Fight Club
Cluster 2	6.1	The Lord of the Rings: The Fellowship of the Ring, The Matrix, Star Wars
Cluster 3	7.0	Shrek, Toy Story, Finding Nemo, Monster Inc.
Cluster 4	6.2	The Princess Bride, There's Something About Mary, The Big Lebowski

Table 1. K-means Clustering Results

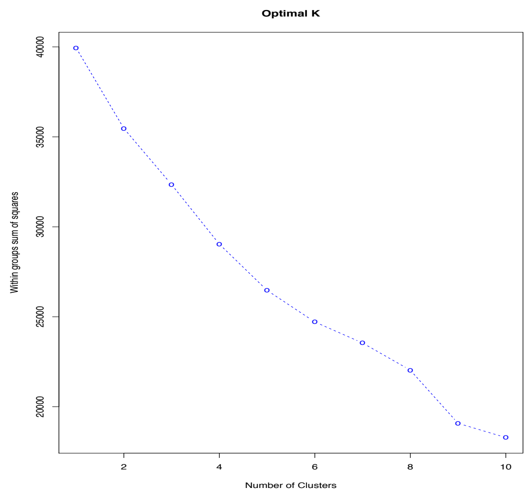


Figure 6. K-means Optimal K Choice

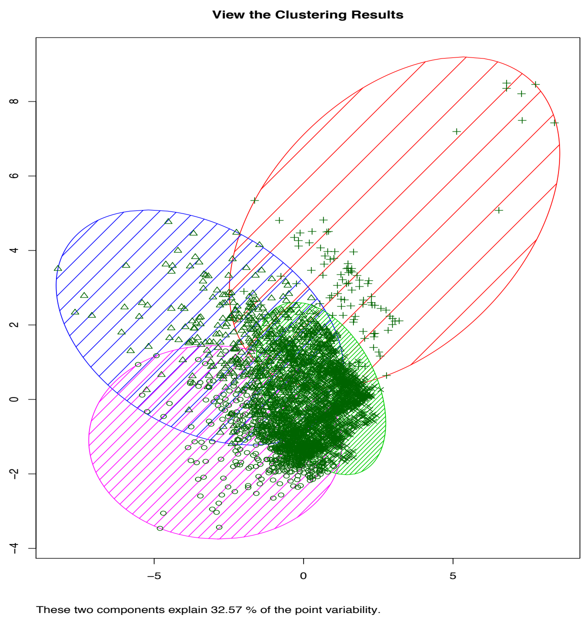


Figure 7. K-means Visualization on 2-Dimensions

In DBSCAN, a weighted version is utilized. The weight for each data point is calculated as $\frac{Votes}{2010-Year}$. By choosing clusters with a relatively more points, we have 6 clusters (shown in Table 2). By examining some representative movies in each cluster, we can infer that the first cluster has the most Oscar nominated or awarded movies (e.g. "*Fight Club*", "*Se7en*". The cluster 2 includes commercial or scientific movies (e.g. "*Spider Man*", "*Jurassic Park*"). The cluster 3 includes terror or crime movies (e.g. "*Reservoir Dogs*", "*The Shining*"). The cluster 4 contains action movies (e.g. "*Kill Bill*", "*The Rock*". The cluster 5 are generally some good movies, but may not be as famous as in first cluster (e.g. "*Lost in Translation*", "*Rain Man*"). The last cluster contains movie with genres are comedy or black humor (e.g. "*American Pie*", "*Snatch*").

Clusters	Average Ratings	Examples
Cluster 1	7.1	The Sixth Sense, Memento, Se7en, American Beauty
Cluster 2	5.8	Spider Man, Die Hard, Jurassic Park, Indiana Jones and the Last Crusade
Cluster 3	6.2	Reservoir Dogs, Psycho, The Shining, Jaws, Alien
Cluster 4	6.5	Heat, The Rock, The Minority Report
Cluster 5	6.7	Chicago, Lost in Translation, Rain Man, Being John Malkovich
Cluster 6	5.8	Dogma, American Pie, Snatch, O Brother, Where Art Thou?

Table 2. DBSCAN Clustering Results

Sentiment Analysis

According to sentiment analysis result, people tend to have a positive attitude toward movies as the average sentiment is 0.2 (visualized in Figure 8). Among all different movie categories, Drama dominates all other movie types. Musical, sports and horror movie has far less production (visualized in Figure 9). However, having more movies in one category does not necessarily mean audience or critics will like it. And science fiction is most favorited by critics and audience (visualized in Figure 10).

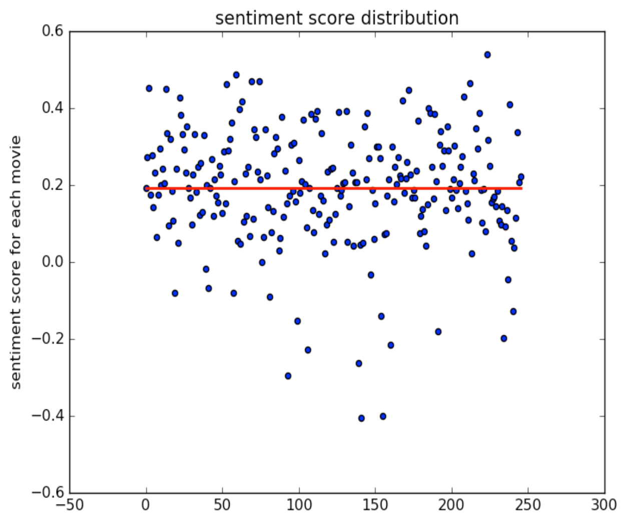


Figure 8. Sentiment Score Distribution

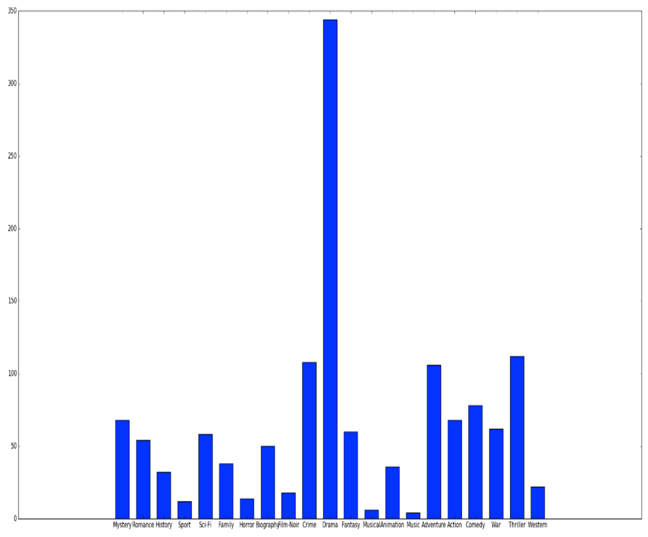


Figure 9. Movie Category Distribution

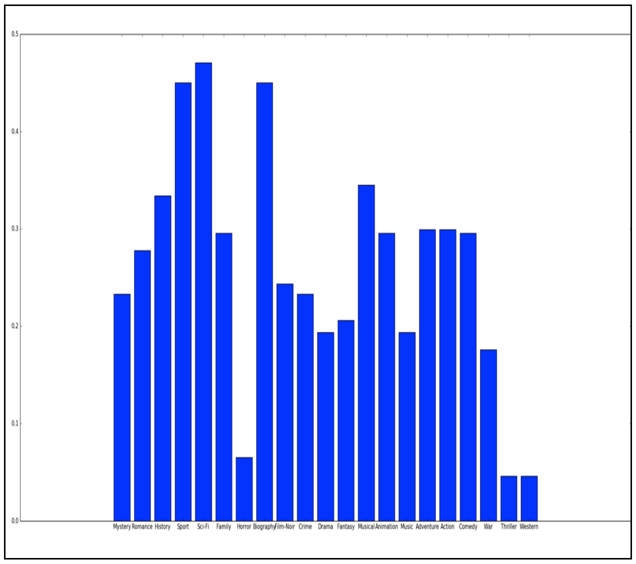


Figure 10. Category Sentiment Score

At last, we also did word cloud examples on "*The Zootopia*" and "*The Revenant*". Word cloud provides a great way to visualize key words in the movie so that people could have a general sense of what a movie would be like or what are related to the movie topics without even watching it (visualized in Figure11,12).

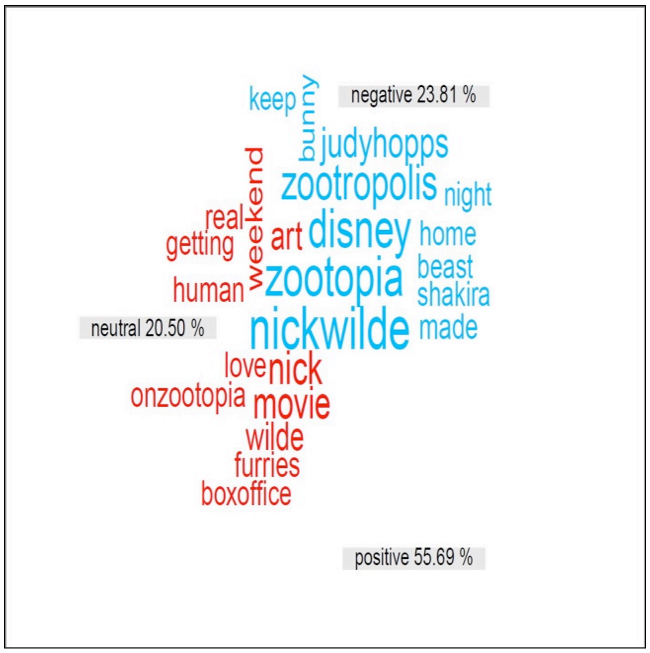


Figure11. Word Cloud (The Zootopia)

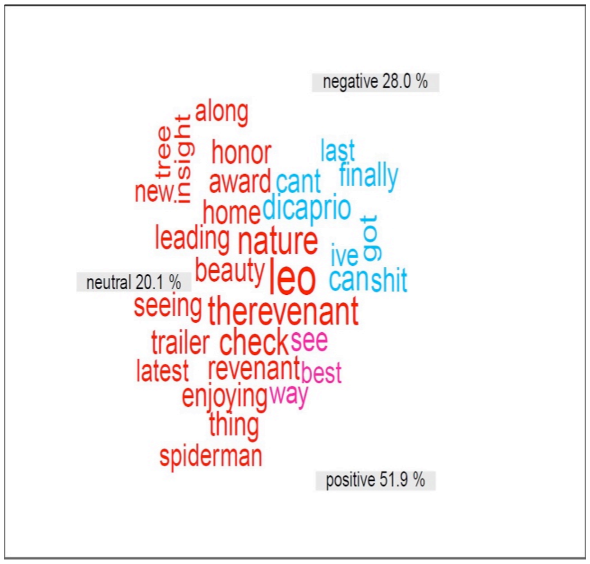


Figure11. Word Cloud (The Revenant)

VII. Conclusion

SQL language is pretty quick in handling the dataset with huge volume of rows (not necessarily in our case). To conduct statistical analysis or learning algorithm, R packages offer a lot benefits. Python and R are both good tools to scrape and collect web data.

Supervised learning methods indicate better prediction (less variance and better precision) among higher rating movies. This means we can use the built features such as budgets, movie length plus an estimated vote to predict movie rating and do recommendation for users. The estimated votes can be collected through internet poll or being approximated using statistical sampling methods.

Clustering methods perform a reasonably good job in separating these movies based on genres and ratings, which also could be a good start point for movie recommendation system. It can give a recommendation to users by comparing respective cluster with users' previous preference clustering results. We can even use cluster results to roughly predict movie ratings. Therefore, in future work, users' previous rating and like might be a major data source.

As we are more interested in good movies (especially top movies). A sentiment analysis would be useful for us to better understand the users' need.

Acknowledgment

The authors would like to sincerely thank Professor Lin and all TAs for their kindly help and great suggestions with all course materials and the final project.

References

[1] http://blog.secaserver.com/2013/08/importing-imdb-sample-data-set-mysql/

[2] http://www.r-bloggers.com/create-twitter-wordcloud-with-sentiments/

[3] https://github.com/JulianHill/R-Tutorials

[4] http://stackoverflow.com/questions/1793532/

[5] http://www.r-bloggers.com/predicting-movie-ratings-with-imdb-data-and-r/

[6] http://www.r-bloggers.com/top-250-movies-at-imdb/

[7] https://rpubs.com/arun_infy13/97529

[8] IMDBpie: https://github.com/richardasaurus/imdb-pie

[9] Alchemy API: http://www.alchemyapi.com/

[10] Corresponding R package CRAN repositories ((sqldf, e1071, randomForests, dbscan, cluster, wordcloud, twitteR) and Python Package Indexes (imdbpie)

[11] Information courtesy of IMDb (http://www.imdb.com). Used with permission.