CS591 Final Project report

Recommendation System for Amazon Games

Leshen Sui, Zheming Sun

12/12/2016

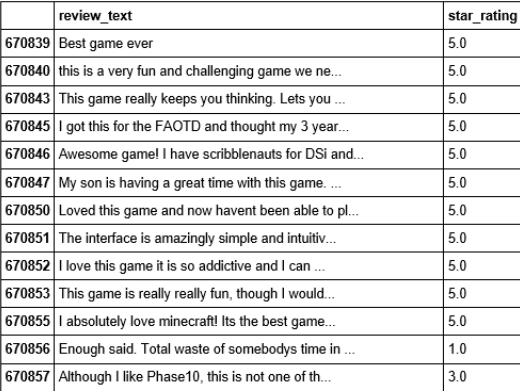
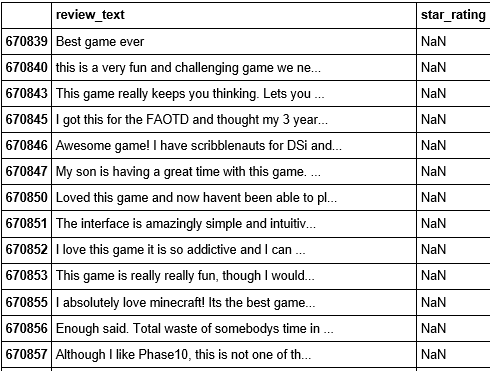
**Abstract**

In this project we are aimed to build a recommendation system for Amazon game apps. For users who give a high star-rating on a certain app, we are going to do recommendation based on two approaches. The first approach is based on clustering, which the goal is to recommend users similar apps based on their description and the average star-rating. The second approach is based on matrix completion(fill in a user-item matrix based on known entries). The expected outcome is the user will give a high star-rating on our recommendation.

**Dataset and data preparation**

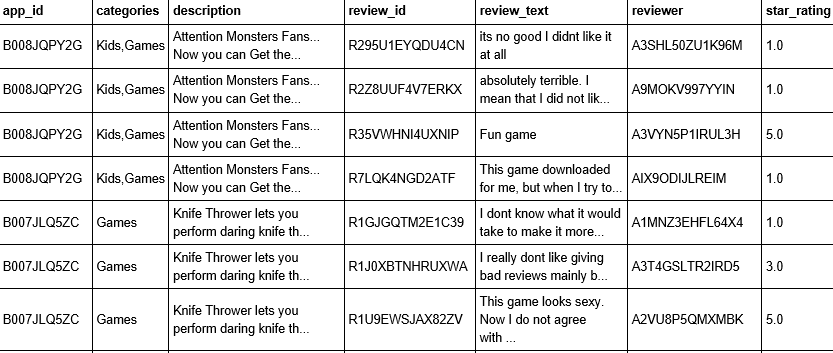
Our dataset is a subset of a huge Amazon app dataset that contains information about apps and users and meta-data about them, including description for apps, categories of apps, developer, reviews, star-rating, review time...For our project purpose, we select all apps whose category is related to game, and keep six columns that potentially useful, which are “asin”(app id), description, reviewer id, review text, user id, and star rating. We notice that some of the star rating is missing, so we use classification techniques to predict the missing values by stemming, removing stop-words and TF-IDF vectorizing reviews. First we pick out all records with non-missing star-rating, and then split those records into training set and validation set(7:3) and tried different classifiers. After fitting the model we do evaluation based on the average difference between the true star\_rating and predicted staring\_rating. The intuition is we want our predicted star\_rating is close to the true star\_rating; for example, predicting a 5 star to a review when the true rating is 4 star is acceptable, but we want to avoid that predicting 1 star rating as 5 star. Therefore avg(|ytrue - ypredicted|) should be as small as possible. Among our experiments, Naive Bayes, Random Forest, Linear regression and Logistic regression gives the final outcome; SVM and KNN are too slow so I give them up. Here is the result:

It turns out that logistic regression gives highest accurate score and lowest average error, The average error of Logistic regression model is 0.58, which means that if I predict a star\_rating as x star based on logistic regression model, the true value should fall in the range (x - 0.58, x + 0.58) on expectation, which is acceptable result. Also, the accuracy score is 0.65, which means 65% of the test data are prediction exactly correct. Therefore, we are going to use logistic regression to predict our missing star-ratings:



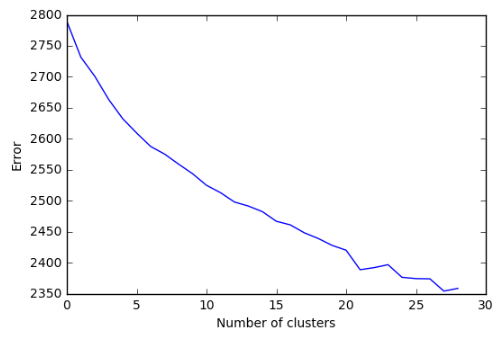
Notice that most of the predictions make sense. For example, “Best game ever” is predicted as 5 star, but “total waste of somebody’s time” is predicted as 1 star.

Our final dataset looks like the following. There are 7896 games and 351387 users in total.

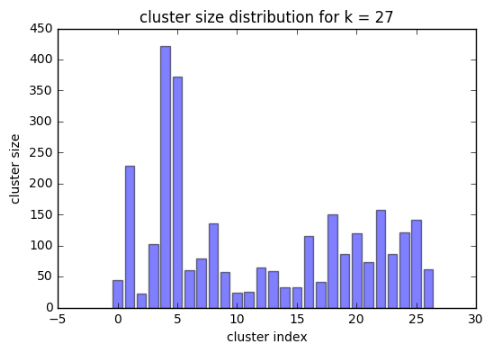


**Recommendation based on clustering**

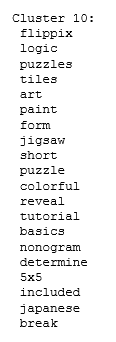
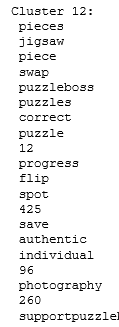
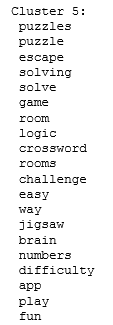
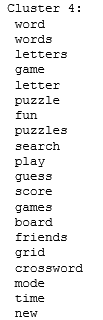
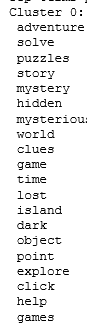
Our first approach is to cluster games on their descriptions, because game with similar descriptions may belong to same types. For example, if a reviewer rates highly for FIFA 17, then he might be a sports fan so recommending good sports game to him would be reasonable. Again, we use some normal text processing like stemming and TF-IDF vectorizing to construct a document-term matrix, and apply k-mean clustering algorithm. The first step is to choose a reasonable k value. In order to achieve this we utilize “elbow method” to find the k value that minimize the total error. It turns out that k = 27 is the best choice.



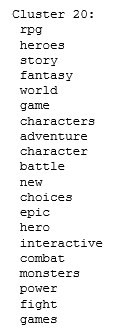
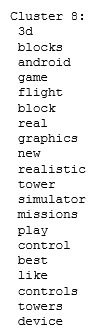
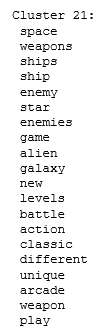
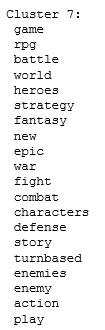
It’s ideal that all clusters have similar number of games, so we plot the distribution of clusters’ sizes. Most of the clusters have a size between 50 and 200.



Then we look deep into each cluster, trying to assign some meaningful labels for them. First we find some clusters that related to puzzles and words game:



We also find some RPG games related to war:



Other types of games involving trivia games, animal hunting games kid games and cards games. Most of clusters have top terms that indicate game types. Therefore, we are pretty successful in clustering similar descriptions together. Then, we are going to rank apps in each cluster by the average star-rating. The top-10 apps are what we are going to recommend to users if they give a 5 star-rating to an app in the same cluster.

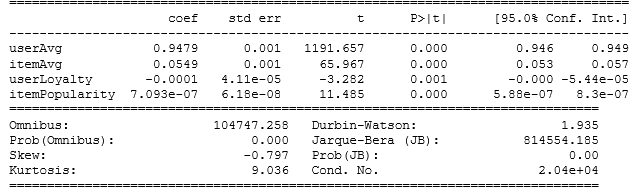
The advantages of this recommendation algorithm lie in two aspect: first, the games we recommended are similar in types. We’ll avoid the case that a reviewer provides a high star rating on sports game but we recommend him angry bird. Second, the games we recommended have high quality. Only games that have high star rating will show up on the recommendation list.

However, this recommendation approach is over conservative. Users are only recommended with games in same categories, and only games that have high overall rating have chances getting recommended. Sometimes a game with relatively low average rating may meet the preference of some special users. For this consideration, we design another approach based on user-item matrix completion.

Utilizing the networkx package, we build a bipartite graph consists of users and games, and a weighted edge represents the corresponding star rating. Our goal is to predict the weights of edges which currently not exist. Intuitively a star rating depends on two aspects: the “harshness” of users in terms of rating and the quality of the item. So we take the average of that user’s past rating and the item’s average rating as two importation features. Besides, we also take consideration of the popularity of a game, reflected by the number of users who reviewed this game; and the loyalty of a user towards Amazon games, reflected by how many times the user wrote reviews. Intuitively, popular games tend to receive high star ratings, and loyal users tend to give high star ratings. Still, we split our dataset into two parts, one part for training(70%) and one for validation(30%). Again, we construct 4 models: Random forest, naive bayes, logistic regression and linear regression. We also take KNN and SVM into consideration, but those two method take too long to generate an output. Here are the performance plot:

In this case we prefer linear regression to other models. On the one hand, the R^2 is almost 0.8, meaning that 80% of variation are explained by current features. On the other hand, linear regression generates lowest MSE value, because that’s the only method which can output decimal numbers(Other methods are classification techniques whose prediction output are categorical, so the result will not be as smooth as linear regression).

Also, we make sure that all fours features are significant(95% CI not cover 0):



Then, it’s able to use the model to compute all missing entries of our user-item matrix. Then, if we predict a user will rate highly to a game, we can recommend the game to that user. This method takes