Finding experts in cellartracker dataset

Jinhui Wu
University of California San Diego
9450 Gilman Drive
San Diego, 92092
jiw241@ucsd.edu

ABSTRACT

In this assignment, I want to identify the experts in the users in the Cellar Tracker data set. To do this, I first go through the data set to understand the properties of the users and the relations between users and their ratings and reviews. Then I tried to give a definition of expert using the result I found from the data set. After that, I use the KNN(k nearest neighbors) algorithm using feature extracted by recommendation model, and I also use the linear regression algorithm using features extracted by the bi-gram model, to classify experts and amateurs. The result shows that the performance of the combination of KNN algorithm and recommendation model is better.

1. INTRODUCTION

When talking about users in a wine review website like CellarTracker, it is an interesting question whether are all the reviewers share the same taste, or is there a difference between experts and amateurs? By going through the data set, if we can find two different behavior patterns, then we can tell the differences between users.

Another question is how to classify a user as an expert or a amateur? This is a classification problem and there are many approaches such as linear regression, support vector machine and k nearest neighbors. From 'case study: beer experts' in the class, we saw a complicated and accurate way to discover experts, but here I want to find a simpler but more efficient way to do it.

2. THE DATA SET

CellarTracker data set is a data set consists of wine reviews from cellartracker. CellarTracker is a website that stores information about wines and wine collections. It has more that 288,000 registered users with 30 million individual bottles, and nearly 3.8 million wine reviews from users. The data set consists reviews from 2003 to October 2012,

WOODSTOCK '97 El Paso, Texas USA


```
wine/name: 2001 Thierry Allemand Cornas Reynard
wine/wineId: 40451
wine/variant: Syrah
wine/year: 2001
review/points: 92
review/time: 1195948800
review/userId: 1
review/userName: Eric
review/text: Fantastic wine! Blackberry, smoke, olive, stem,
floral notes and bit of tar. This is one expressive nose for
a young wine. Saturated palate. Juicy with black cherry notes
playing with tar.Quite acidic and fresh. Very young.
Terrific character on this wine. Surprisingly polished for a Cornas.
```

Figure 1: A sample review

there are 2,025,995 reviews, 44,268 users and 485,179 wines in the data set.

As is shown in figure 1, each review consists of 9 fields, which are the name of the wine, the unique ID of the wine, the variant of the wine, the year that the wine is produced, the rating points, the time of the rating in UNIX format, the unique ID of the users, the name of the user and the review given by the user.

I performed preprocessing on the data set. Some reviews have 'N/A' in review/points or review/text fields, so I create a filter that only keep reviews with valid information in these two fields. There are 556,171 reviews and 7062 users left. Besides, I randomly split the data set into training set and testing set which have the same size.

By going through the reviews, I noticed that different users have given significantly different number of reviews, so I want to know if there is a relation between the number of the reviews and the experts. Then I noticed that the distance between the user's rating and average rating for a given variant is different. So I compute the MSE(mean squared error) of a user between all his ratings and the average rating of each variant. Here I use variant instead of wine name, because there are many different wine names that related to the same variant, and there are only 636 different variants, so it is much easier to handle.

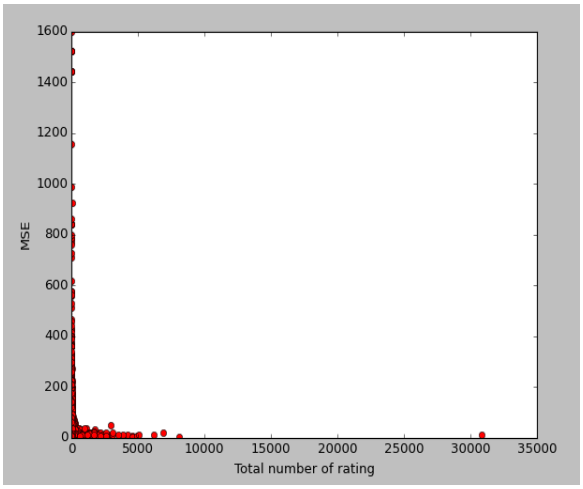

Figure 2: The relation between rating MSE and total number of ratings

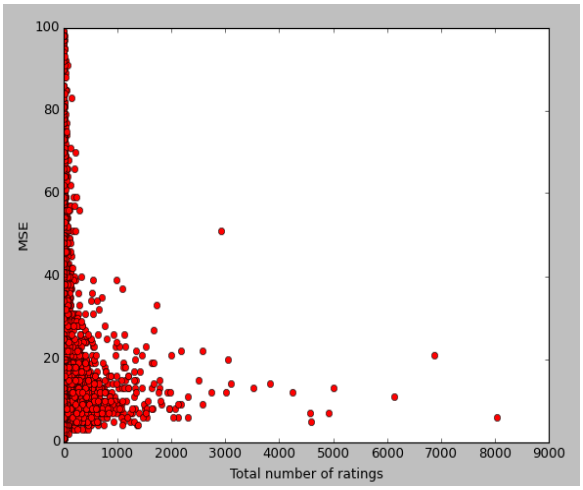

Figure 3: Details of figure 2

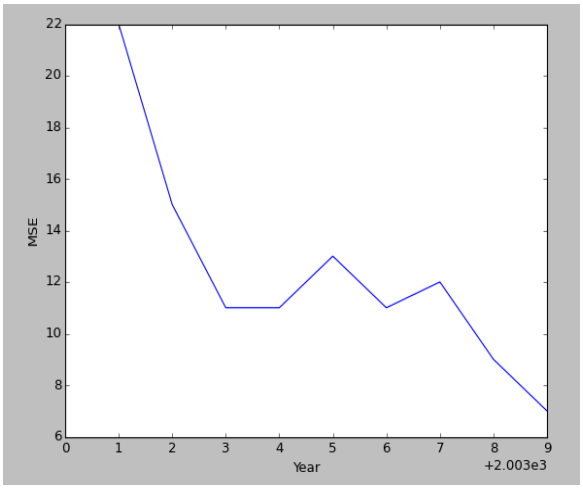

Figure 4: Temporal MSE change for user '70'

In figure 2 and 3, each point represents a user. Figure 4 shows the temporal MSE change of user '70'. From figure 2, 3 and 4, I found that I can use the total rating number and the MSE as a sign to classify an expert.

3. PREDICTIVE TASK

To give a predictive task, I first came up with a definition of expert and amateur. As is shown in figure 5, I choose MSE=20 and total number of ratings=400 as a threshold, all the points located in the bottom right corner are identified as experts, other points are identified as amateurs.

Then the predictive task is a classification problem. I first fit a recommendation model and a bi-gram model to the data set. For the recommendation model, I can get a parameter which represents how much a user tend to rate above the mean. I use this parameter to do KNN(K-nearest neighbors) classification to find experts. For the bi-gram model, I use the bi-gram feature to do linear regression. And the error rate will be used to evaluate the classification.

After preprocessing, each review in the data set contains 9 valid fields, and the information we can use here is the ratings and the reviews given by each user. After consideration, I decided to use two different models, one is the recommendation model so I can use the information provided by the ratings, another one is the linear regression model using the bi-gram as features so I can use the information in the reviews.

The recommendation model was mentioned in class:

$$f(u,i) = \alpha + \beta_u + \beta_i$$

Where u and i represent user and item, and β_u represents how much this user tends to rate things above the mean, while β_i means how much this item tends to receive higher ratings than others.

Then the optimization problem becomes:

$$argmin_{\alpha,\beta} \sum_{u,i} (\alpha + \beta_u + \beta_i - R_{u,i})^2 + \lambda[\sum_u \beta_u^2 + \sum_i \beta_i^2]$$

To solve this problem, we update each parameter in the following way until it converges:

$$\alpha = \frac{\sum_{u,i \in train}(R_{u,i} - (\beta_u + \beta_i))}{N_{train}}$$

$$\beta_u = \frac{\sum_{i \in I_u} R_{u,i} - (\alpha + \beta_i)}{\lambda + |I_u|}$$

$$\beta_i = \frac{\sum_{u \in U_i} R_{u,i} - (\alpha + \beta_u)}{\lambda + |U_i|}$$

Then we can use β_u to classify a user as an expert or a amateur.

The linear regression algorithm is as follows, where X and y is the bigram feature of the data and the label, respectively. And θ is the weight, while b is the bias.

$$y = X\theta + b$$

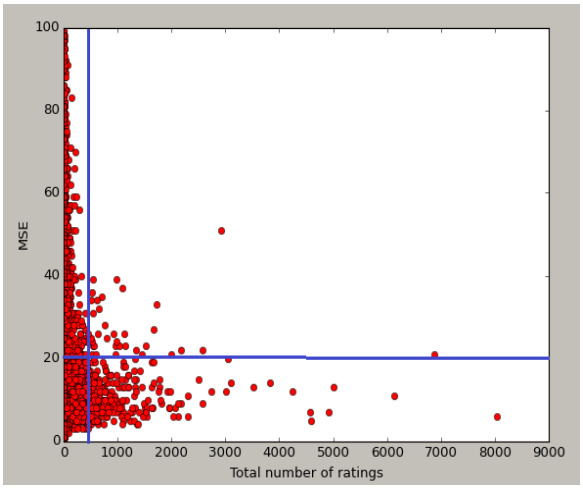

Figure 5: Definition of experts

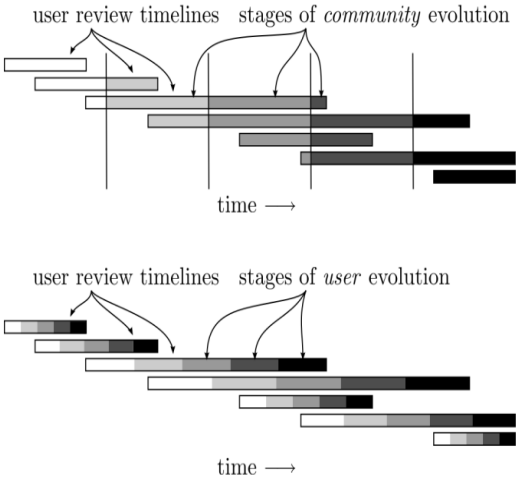

Figure 6: User evolution

The KNN algorithm is to classify a data to the same label of the majority of its k nearest neighbors.

There are other approaches such as non-linear regression, but the result of linear regression is good enough, so I didn't use non-linear regression for this task.

4. RELATED LITERATURE

The Cellar Tracker data set is originally used for[1] and is from SNAP. The Ratebeer and Beeradvocate data set have been studied in similar ways. The paper focus on how to effectively characterize acquired tastes and expertise. The paper based on the observation that 'people evolve and develop at different rates'. So it tried to learn the rate of development for each user. It replaced the 'standard' model:

$$rec(u,i) = \alpha + \beta_u + \beta_i + \gamma_u * \gamma_i$$

With one whose parameters change as a function of time(t):

$$rec_t(u,i) = \alpha(t) + \beta_u(t) + \beta_i(t) + \gamma_u(t) * \gamma_i(t)$$

The outcome is great: it has 6% , 13% and 23% improvement on beer, wine and movies(Amazon) prediction. And it

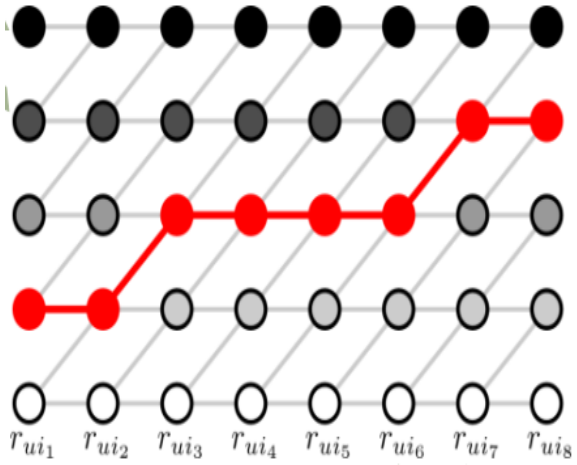

Figure 7: Models of user and community evolution

can also model other types of data including medical records. The paper also used the conception of expert. It revealed that experts are more predictable than beginners and they are more inclined to agree with each other. However, I use the same conception of expert in this assignment with a simpler definition which is discussed in the section 'predictive task'.

5. RESULT

5.1 KNN CLASSIFICATION

First I fitted the recommendation model to the training set.

In the model described above, β_u represents how much this user tends to rate above the mean. This is a one dimensional data, to perform KNN algorithm, I added one dimension to each data, so the i-th user is represented by [β_{ui},1].

I tried different configuration of k, and the result is shown in figure 8.

We can see that when k equals 1, we can achieve the lowest error rate, which is 0.0016, and the error rate increases as k increases.

5.2 LINEAR REGRESSION

There are 2,396,978 unique bigrams in the reviews.After I filtered out the stopwords, there are 1,670,038 unique bigrams remained.

Then I use the first 100 bigrams as the feature of each review and perform linear regression on the training data.

After that, I use the result of linear regression to predict if a user is expert on the testing data. The error rate is 0.3666.

5.3 Conclusions

By comparing the error rate, the performance of the combination of recommendation model and KNN classification is much better than the other approach.

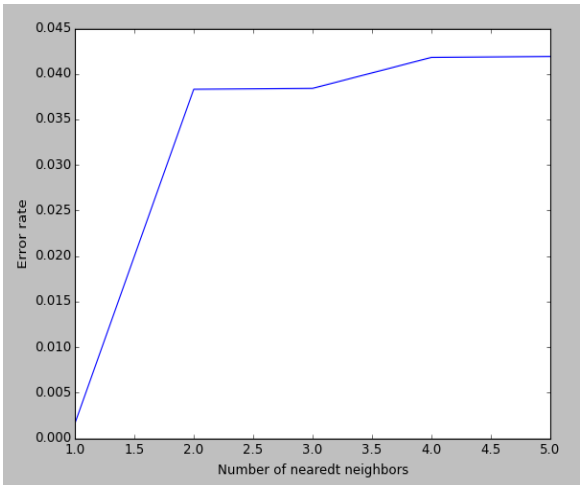

Figure 8: Error rate for different K in KNN classification

When we use 1 nearest neighbor model, the error rate can be as low as 0.0016.

6. REFERENCES

[1] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *World Wide Web*, 2013.