Predicting User Ratings Using Status Models on Amazon.com

Borui Wang
Stanford University
borui@stanford.edu

Guan (Bell) Wang
Stanford University
guanw@stanford.edu

Zhemin Li
Stanford University
zhemin@stanford.edu

Group 19

1. ABSTRACT

Amazon is the world's largest online retailers where millions of customers purchase and review products on its website. However, many Amazon customers do not review and rate products after the purchase, and tons of research projects work on producing better prediction strategies for customer ratings. In this research project, we show a new approach to enhance the accuracy of the rating prediction by using machine learning methods that learn from a graph based on user status, defined by features such as the helpfulness votes and total votes received by the users. We discussed how the training performance of our model changes as we change the training method, the dataset used for training and the features used in the model. We compared this graphical model with another non-graphic model that is also based on customers's review and status with different settings, and we achieved over 95% prediction accuracy using our graphical status model.

2. RELATED WORK

Predicting user ratings is one of the core issues in recommendation systems. The relevant works in recommendation systems mostly focus on enhancing the quality of the algorithms to increase the prediction accuracy. The most popular system [2] for rating prediction is collaborative filtering, which seeks to predict the expected rating a user may give to an item he/she hasn't rated yet. Memory-based and model-based (e.g. matrix factorization) [3] are the two most popular methods for collaborative filtering. However, both systems take a sparse user-product rating matrix as an input and may fail to capture the quality of the users and their reviews. In a similar project "Using Properties of the Amazon Graph to Better Understand Reviews" [6] proposed by Leon, Vasant, Sheldon (2011), the predicted feedback score was represented with a linear combination of the following features: Original Rating, Review Helpfulness Count, Review Unhelpfulness Count, Review Helpfulness Ratio, Reviewer Helpfulness Ratio, Reviewer Bias, Reviewer Expertise, Reviewer Clustering, Reviewer Betweenness, and Reviewer Degree. However, the goal of their project is to predict the rating of a customer who reviews the same product more than once, while our project is to predict the user's first-time rating score. Also, they use stochastic gradient descent to minimize the mean square error to capture the weights of such features, while we tried more than five different learning methods and analyzed their results in our project.

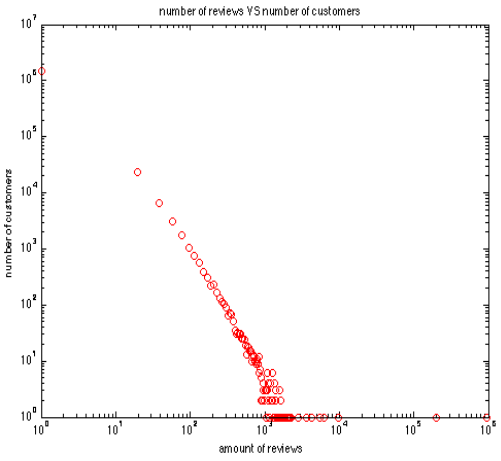

Figure 1: number of reviews v.s. number of customers

3. DATA COLLECTION PROCESS

We used the Amazon product co-purchasing network metadata from project SNAP[1] for our research. The dataset includes over 7 million review scores and helpfulness votes by 1.5 million customers for 0.5 million products. To convert the Amazon meta data into the format we need, we developed a Ruby preprocessing program for data parsing. For the non-graphical model, we parse the data into matrix files readable by Matlab and csv files for NetworkX libraries in Python. Concretely, the output uses row entries to describe which products are reviewed by which customer and their rating scores. For the graphical model, we parse the data into hash structures representing customer-to-product and product-to-customer relationships based on the chosen dataset size to build the graph.

4. INITIAL FINDING AND STATISTICS

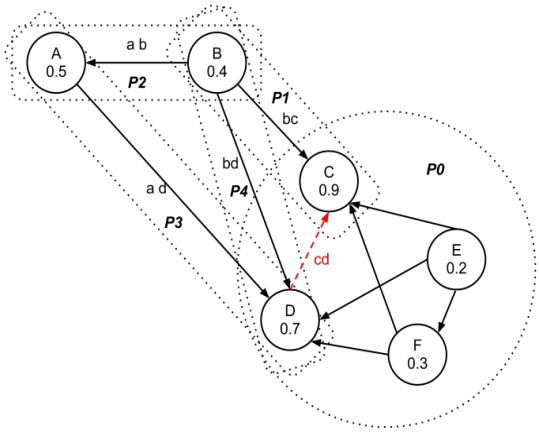

Figure 2:

Because using the entire dataset to build the graph for our status model turned out to be slow on NetworkX, we can only use a subset of all the data for training. We started with using a graph built from 1193 customers who reviewed 200-400 products to reduce the path searching complexity in our graphical status model (see figure 1. above for the number of reviewers vs number of customers, which appear to be a nice power law distribution). Such range of review counts also happened to give the constructed customer graph a high average cluster coefficient of 0.55. We discussed our training result using such dataset in section 5.4. In later sections (5.5), we show that selecting customers of different ranges (100-150, 150-200, 400-600, 600-800) of review counts affects the graph cluster coefficient and ultimately affect our learning accuracy.

5. GRAPHICAL STATUS MODEL

5.1 Overview of the graphical status model

Here we describe the details of the graphical status model. The model assumes that the rating which user U gives to a product is related to the status of him/herself and the statuses of the other users who also happen to review the same products as U. The status of an user in our model is defined as the total helpfulness votes divided by the total amount of votes he/she received on Amazon. Under such definition, the status of an user has the inclusive range [0, 1]. To see how a user's status can be related to other users' statuses, consider figure 2. The solid circles A, B, C, D, E, F represent 6 different users and their statuses in decimal format, and the shapes P_0, P_1, P_3, P_3, P_4 with dotted border represent 5 distinct products. The circle representing a user U is placed within the shape representing the product P if U reviewed P. For example, user A , D reviewed product P3; customer C, D, E, F reviewed P_0. We create edges between customers who both reviewed the same product, and the direction of the edge depends on the difference between the users' statuses. As shown in the figure 2, the edge always points to user of higher status. We try to observe if there is any relationship between the statuses of related customers and their reviews. We utilize the user statuses and co-review relationships in the graph to relate the users. Assume we want to predict the rating of P_0 given by customer D with all the edge and user information in the graph. We first find out if there is any outgoing path from D to another customer in P_0 in the graph, where the path itself does not pass through an edge in P_0. If such path exists, we find the shortest path and record the attributes of the customers along the path to construct the machine learning objectives (introduced later) to predict the rating D given to P0. In the figure 2 above, the shortest path $D \to B \to C$ was chosen. Note that edge $D \to C$ is not valid as it is in P_0. The length of the path we could find depends on the co-review patterns. We expect the path to be short if two customers who reviewed different products can be connected by co-review relationships with the same customers. We show later that over 97% of the customers in our selected dataset could find such path of length two, which forms triangles of product co-review pairs such as triangle ABD and BCD.

5.2 Triangle types and learning features in the Graphical Status Model

In order to build confidence of using the graphical model, we explored the abundance of co-review triangles as defined previously in the graph. We used NetworkX for python to generate the customer relationship graph using customers who reviewed 200-400 products, and found that there are 1193 nodes and 27,9147 edges in the graph. The graph is highly connected, and the largest connected component is the graph itself and the average clustering coefficient is 0.55. As a result, when we use BFS to search the paths as defined previously for the customers, we find that 97.14% of all paths have length two. Thus the graph contains a large number of triangles co-review relationships. This property supports our model and gives sufficient amount of instances to form a dataset for training and testing our model. In order to apply the concept of status, we assign each customer with a status score. We tested several means of computing a customer's status based on statistics such as the total number of reviews, total helpfulness, total number of votes, and total rating scores from the customers. One simple definition of status is the percentage of helpful votes, which is total helpfulness divided by total number of votes. After assigning the status score to each customer, we can build the graph with signed edges. For example, if A's status is higher than B's, then edge AB is positive, otherwise AB is negative. Thus we can have eight different triangle types, where the triangle type will be used as one feature in our machine learning process. Later, we found that there is usually more than one triangle for most customers in our graph and we discussed how the training result changes as we add more triangles in our training features in 5.5.

5.3 Other learning features

Besides the triangle type, we found some other features for machine learning. Take the previous example. We have customer D, B, C and product P_0. Noted that we are going to predict customer D's rating on P_0, and customer C has also rated P_0. The other features we could possibly include in our learning model are therefore D's average rating, C's rating on P_0 and P_0's average rating. The inspiration of choosing these features is that when a customer rates a product, he usually considers the product's average rating which reflects the general public opinion, and his own average rating which reflects his own personal opinion. We also think a customer's

rating can be influenced by his neighbors in the graph, so we added the neighbor's rating on the same product in our learning model as well.

5.4 Initial Learning Results

Each three customers and one product that meet the definition of a triangle as defined previously is a training example in our machine learning model. We used Weka[4] to train and test the dataset. After testing several different machine learning methods, such as linear regression, neural network, KNN, decision table, we found that KNN gave the best performance. We tested our model on a dataset using 7,1542 instances with two test options: cross-validation with 10 folds, and percentage split with 66% training instances and 34% testing instances. Below are the results, where we measured mean absolute error, mean squared error and test accuracy for each case.

	MAE	MSE	Accuracy
KNN	0.1319	0.2546	91.23%
Linear Regression	0.6082	0.8765	50.55%
Neurak Network	0.7896	1.2507	40.29%
Decision Table	0.5632	0.334	54.88%

Table 1: Cross-validation with 10 folds

	MAE	MSE	Accuracy
KNN	0.1883	0.3609	87.46%
Linear Regression	0.6098	0.8765	50.33%
Neurak Network	1.1290	1.7041	15.22%
Decision Table	0.5647	0.8464	55.19%

Table 2: Percentage split with 2/3 training data

From results in table 1 and table 2, we can conclude that KNN gives the best prediction. With cross-validation, KNN correctly predicted 91.23% of testing instances. With percentage split, KNN correctly predicted 87.46% of testing instances. The results indicate that our model is reasonable and effective for predicting customers' ratings on Amazon products.

5.5 Improved Graphical Model

The training methods and training model from the previous section gave good results that verify our intuition of the graphical model. In this section, we improve our training model by adding more triangle paths from our graph. We also have done several experiments with different datasets that represent customers who review different amount of products. The results supported our model well and gave us a good guideline on how to select features for the learning process.

Specifically, our first experiment is designed to check if the model works for different datasets that represent users who review different amount of reviews. The second experiment is to apply the model on one dataset but with different number of triangles representing co-review paths. The third experiment is to measure the performance of the model when the input graph of the dataset is not fully unveiled, for example, by randomly removing customers from the graph. We adopted KNN approach in Weka to perform all the experiments as it is proved to give the best training method from the previous section.

5.5.1 Experiment 1

In this experiment, we applied our training model on five datasets with two triangle modes, where we use 1 and 2 triangle paths from the graphical model as training features. The datasets vary in the range of the amount of the reviews given by the customers. For example, the column "100-150" in the tables below represents the training set of customers who reviewed 100-150 products.

Table 3 contains the datasets' properties, which include the number of instances in each dataset and the average cluster coefficient of the graph generated from the datasets. We found that as the customers in our dataset review more products, the average cluster coefficient of the graph increases indicating that the graph becomes denser.

dataset	number of instances	Average cluster coefficient
100-150	162329	0.3186
150-200	86102	0.3962
200-400	218553	0.5499
400-600	59285	0.744
600-800	8584	0.8523

Table 3: dataset Properties Table

For each dataset, we tested our model with cross-validation with 10 folds. Below are the results trained for different datasets with 1 and 2 triangles as training features. We measured the mean absolute error, mean squared error and test accuracy for each case similar to the previous section.

dataset	MAE	MSE	Accurary
100-150	0.165	0.3125	88.79%
150-200	0.1324	0.2377	90.63%
200-400	0.1756	0.306	87.33%
400-600	0.1324	0.2018	89.48%
600-800	0.0939	0.1612	93.12%

Table 4: Cross-validation with 10 folds (1-triangle mode)

From the table 4 and 5, we can find an improvement of the training accuracy as we use more triangles for training, and the accuracy for datasets with 1-triangle path as

dataset	MAE	MSE	Accurary
100-150	0.1283	0.2515	91.56%
150-200	0.0.0972	0.1832	93.44%
200-400	0.0.1345	0.2456	90.68%
400-600	0.0.0824	0.1383	93.8%
600-800	0.0712	0.1215	94.79%

Table 5: Cross-validation with 10 folds (2-triangle mode)

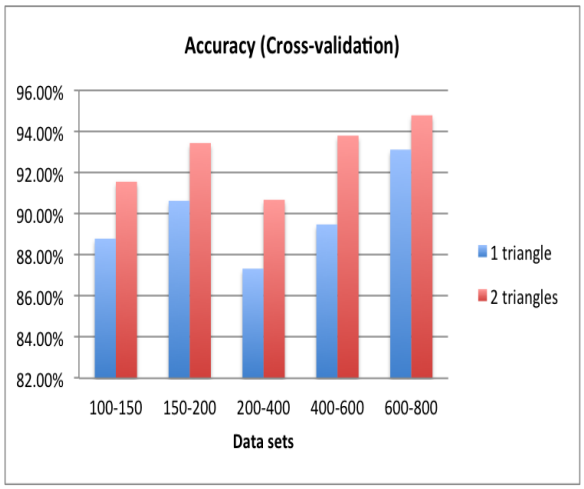

Figure 3:

learning feature is around 90%, and the accuracy with 2-triangle paths is around 93%. This prediction accuracy is very high. In addition, the training results indicate that our model also works for datasets of different ranges. We will show how many triangles gives the optimal results in the second experiment.

5.5.2 *Experiment 2*

In this experiment, we applied the model on the same dataset using different amount of triangles derived from our graphi as features for training. The dataset we used is the one where the customers' number of reviews ranges from 200 to 400. Again, similar to the first experiment, we adopted cross-validation with 10 folds method. The results is in table 6.

From table 6, we can find that using 3 triangles for our training features gives the best performance, which achieved the highest accuracy of 96.63%. Moreover, adding the second triangle improved the performance the most, because the accuracy was improved by 1.93% after changing from 1-

# of remaining instances	MAE	MSE	Accuracy
1	0.0914	0.1853	94.19%
2	0.0617	0.126	96.12%
3	0.0556	0.1173	96.63%
4	0.0555	0.1176	96.62%
5	0.0544	0.1129	96.62%

Table 6: Cross-validation with 10 folds (1,2,3,4,5-triangle mode)

# of remaining instances	MAE	MSE	Accuracy
20K	0.6135	1.1828	59.38%
40K	0.5128	0.9811	65.92%
60K	0.4272	0.8123	71.47%
80K	0.3573	0.6734	75.97%
100k	0.3115	0.5831	78.91%
160k	0.2002	0.3729	86.38%
210k	0.1345	0.2456	90.68%

Table 7: Cross-validation with 10 folds (2-triangle mode, 200-400 dataset)

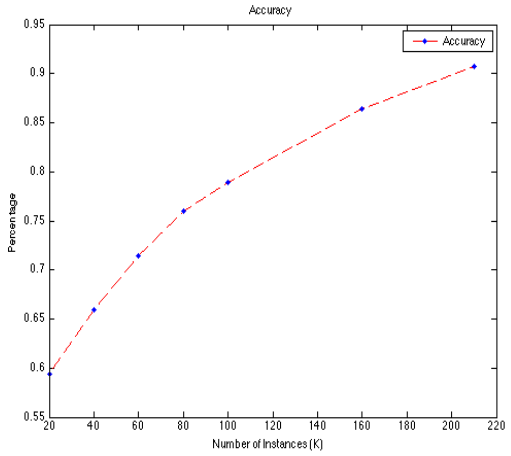

Figure 4:

tiangle mode to 2-triangle mode. Besides these two points, we can also conclude that after having 3 triangles as training features, adding more triangles does not improve the performance. Therefore, the optimal number of triangles for our model is 3.

5.5.3 *Experiment 3*

In this experiment, we tested our model with incomplete data sets to find the relationship between prediction accuracy and the completeness of the data. We randomly removed certain amount of nodes (customers) and edges (co-review relationship) connecting them from the graphs generated from the datasets (200-400). The results are in table 7.

From table 7 and figure 4, we can find that our model highly depends on the completeness of the graph. The model performs badly when a large amount of nodes are removed. This can be explained by the fact that our model uses KNN to produce prediction. This can be explained by the fact that if customers with similar features are removed, the target customer whom we want to predict will not be able to find co-reviewers in the training dataset, therefore hurting the training quality and decreasing the prediction accuracy.

6. REVIEW-BASED STATUS MODEL

6.1 Overview of the Review-based Status Model

In this part of the project, we explore the relationship between the reviews and their influence on customers' future

ratings. Our training samples are based on the reviews of each product. Since the number of reviews varies for different products, we selectively pick K reviews of a product for each training sample. The value of K we chose is 10 and 30 because we observed that products receive such numbers of reviews most often. The order of the K reviews can be based on many criteria. In our project, we chose K most recent reviews for a product. We experimented both K values in the evaluation stage.

6.2 Baseline

In the baseline model, we only considered the impact of the selected reviews for a product to predict ratings. Our assumptions for the baseline model is:

$$R_{Pridict} = \sum(\theta_1 * H_i + \theta_2 * P_i + \theta_3 * V_i) * R_j, j = 1, 2, ...k$$

where the subscript j is the j^{th} customer, P_i is the number of product reviews, H_i is number of helpfulness votes, V_i is the total number of votes. The mean square errors we tested on this model is 3.37.

6.3 Improved Model

We have two different assumptions with the review data. 1) Assuming that the quality of the review is the key influencer to the users; 2) Assuming the reputation of the reviewer is the key influencer to the users. For the first assumption,

$$R_{Pridict} = \alpha * R_{proAvg} + (1 - \alpha) * R_{userAvg} + \sum(\theta_1 * H_i + \theta_2 * V_i + \theta_3 * H_i/V_i) * (R_i - R_{proAvg}), j = 1, 2, ...k$$

where H_i is the number of helpfulness and V_i is the number of votes of the review j. H_i and V_i are the data to measure the quality of the review. The mean squared error (MSE) between the predicted rating using this model and the true rating is 0.75. To find the best model, we tried to use square and cubic of H_i and V_i, but we got higher testing error. However, when we tried H_i/V_i, the MSE decreased by 0.04, which means the ratio H_i/V_i is a valuable information for the quality of the review. In the next section, we will discuss more about the ratio H_i/V_i, such as including its square and cubic forms, subtracting by its average to distinguish between high quality reviews (with positive numbers) and low quality reviews (with negative numbers). For the second assumption,

$$R_{Pridict} = \alpha * R_{proAvg} + (1 - \alpha) * R_{userAvg} + \sum(\theta_1 * H_{i,u} + \theta_2 * V_{i,u} + \theta_3 * R_{i,u}) * (R_i - R_{proAvg}), j = 1, 2, ...k$$

where u is the user of the $i^(th)$ review, and H_i,u is the total number of helpfulness of u ,V_i,u is the total number of votes, and R_i,u is the total number of ratings from the user u. V_i,u, R_i,u and H_i,u are the data to measure the stats of the user. The MSE between the predicted rating and the true rating in this model is 0.75. To find the best model, we tried to use square and cubic forms of H_i,u ,R_i,u and V_i,u, but we got higher testing error. One interesting fact is that when we try to include the feature H_i,u/ V_i,u, there is no improvement in training error.

6.4 Combination: Review-based status model and Graphical Status Model

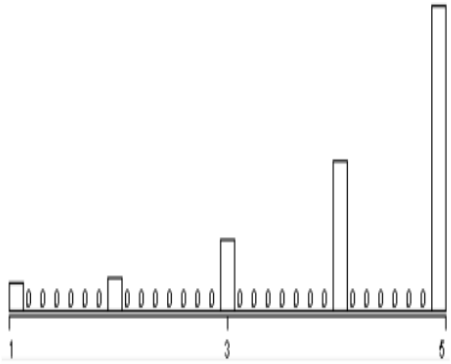

Figure 5: Distribution of ratings in training data

Improved Model 2	K=1	K=5
Mean Square Error	1.4393	2.3592
Absolute Mean Error	0.6977	0.4876
Root Mean Square Error	0.70	0.75
Accuracy	73%	56%

Table 8: Performance Laerned through KNN

Based on the Improved Model 2, we incorporate features used in the Graphical Status Model in the prediction of the ratings. define W_1, W_2 and W_3:

$$W_1 = (R_i - R_{proAvg}) * \sum(\theta_1 * H_{i,u}), j = 1, 2, ...k$$

$$W_2 = (R_i - R_{proAvg}) * \sum(\theta_2 * V_{i,u}), j = 1, 2, ...k$$

$$W_3 = (R_i - R_{proAvg}) * \sum(\theta_3 * R_{i,u}), j = 1, 2, ...k$$

Adding W_1, W_2 and W_3 is mathetically equivalent to include $\sum(\theta_1 * H_{i,u} + \theta_2 * V_{i,u} + \theta_3 * R_{i,u}) * (R_i - R_{proAvg}), j = 1, 2, ...k$ in the learning algorithm. We append W_1, W_2 and W_3 to the extracted feature vector in Graphical Status Model. Our hope was that we were able to provide the learning algorithm more information by combining graphical status data and review-based data. The learning algorithms should be capable of figuring out the best combination of the features by adjusting the coefficient of each feature.

6.5 Learning Methods and Evaluations

We adopted the cross-validation method in this experiment. The mean value for the target value in the training data is 4.139, and its standard deviation is 1.132. The distribution of the ratings in our training data is shown as following, indicating that rating 4 and 5 appear most frequently, which is shown in figure 5.

Since the hypothesis is a combination of weights and scales for our Baseline Model, Improved Model 1 and Improved Model 2, we chose multilayer neural networks as our learning method in our training stage. We use the Mean Square Error (MSE) for performance evaluation. The result is table 8.

Combined Model	Mean Square Error	Absolute Mean Error	Root Mean Square Error	Accuracy
Decision Tree	0.7197	0.5056	0.8484	58.56%
KNN(K=1)	0.6853	1.3224	1.1500	54.17%
KNN(K=2)	0.6443	0.9906	0.9953	49.89%
KNN(K=3)	0.7771	0.5491	0.8815	54.97%
Linear Regression	1.0192	0.6849	1.0095	45.35%
Least Median Square Linear Regression	1.0192	0.6849	1.0095	45.35%
Locally Weighted Linear Regression	1.2386	0.8068	1.1129	40.91%
Multilayer Perceptron 1 Hidden Layers	1.1948	1.0931	1.0931	55.37%
Multilayer Perceptron 2 Hidden Layers	1.1775	0.6481	1.0851	55.43%
Multilayer Perceptron 3 Hidden Layers	0.6454	1.1677	1.0806	55.45%

Table 9: Eveluation result for combined model

Neural Networks	TrainingMSE	Testing MSE
Baseline	3.27	3.33
Improved Model 1	0.73	0.75
Improved Model 2	0.70	0.75

Table 10: Comparision of three models using Neural Networks

From the evaluation results, we find that the baseline model has a very high bias for both the training and testing data, but the MSE for both of them differ little between the training and testing data, indicating a low variance in the learning model. From the previous experiment, the hypothesis in Improved Model 2 has a better performance than other two models. We also tried other learning methods for Improved Model 2, and find that KNN has a decent performance, which is shown in table 10.

For the combined model, we tried four different learning methods: Decision Tree, K-Nearest N (KNN), Linear Regression and Multilayer Perceptron. The results for these learning methods are shown in table 9. We conclude that, for the combined model, KNN gives the best prediction in terms of Mean Square Error (MSE), while Decision Tree, KNN and Multilayer Perceptron gives similar prediction accuracy, that is about 55%. When comparing the results of KNN with combined model and graphical structure model, we find the combined model has worse performance. The reason can be explained by the different interpretations of review-based model and graphical model. Concretely, the graphical model is designed to explore the network path structure of the data, while the review-based model relies more on the linear combination of common user features. Simply combining data from the two models does not give a better result because there is no direct relation between the underlying meanings of the feature vectors from the two models; as a result, the performance of KNN with combined model is worse than graphical model. When comparing the results of multilayer perceptron with combined model and improved model 2, the improvement is not significant. Thus, we can conclude that combining these two data sets do not help.

7. DIFFICULTIES

It is very difficult to model a customer's rating behavior properly. The correctness of a model highly depends on what kind of information the dataset provides and the structure of the model. Hence, we need more data to provide additional information. Besides this, we think it is necessary to take some sociology and social psychology theories into consideration when building the model so that we can build a deeper understanding in this topic.

8. CONCLUSION AND FUTURE WORK

We have explored two models to solve the problem. The first model relies more on analyzing the network structure and using status theory. The second one focus more on review-based status information. From the results, we find the first model gives us better performance, which achieved a mean square error around 0.1173 and prediction accuracy around 96.63% with three triangles used as training features. The second model gives the prediction accuracy around 73% and mean square error around 0.6853. In conclusion, applying network and status concepts in a graphical model is very useful for improving prediction accuracy for Amazon data. Other customers' ratings in a product might produce bias to the target user's rating [5], which is not considered as a parameter in the graphical status learning model. To extend our work, it might be helpful to include parameters reflecting the bias caused by other reviews in the same product to improve our model and reduce learning errors. Furthermore, the definition of user status in the graphical model can be overly simple and naive. Other definitions of user status would change the triangle type and drastically affect the learning result, which is left to be explored. During our experimentation of feature selection, the features selected in the graphical model are not proved to be the best choice of features, and it requires more analysis to come up with a better set of features.

9. REFERENCES

[1] *SNAP*, http://snap.stanford.edu/.

[2] Paul Resnick, Hal Varian. Recommender systems, *Recommender Systems*,Communications of the ACM, 1989.

[3] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handboo*,Springer, 2011.

[4] *Weka is a collection of machine learning algorithms for data mining tasks*, http://www.cs.waikato.ac.nz/ml/weka/.

[5] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, L. Lee, *How opinions are received by online communities: A case study on Amazon.com helpfulness votes.* ,In Proc. ACM WWW, 2009.

[6] Leon, Vasant, Sheldon, *Using Properties of the Amazon Graph to Better Understand Reviews*