Text-Based Rating Predictions on Amazon Health & Personal Care Product Review

Weikang Chen
wec085@eng.ucsd.edu

Chihhung Lin
chl584@eng.ucsd.edu

Yi-Shu Tai
yitai@eng.ucsd.edu

ABSTRACT

In this assignment, we are going to predict the ratings based (mainly) on review texts and find out what kind of words have possitive and negative effect on ratings. We assume a situation that we only get the reviewers' id, products' id, review texts for each reviewer and product pairs and the time the reviews were written. These are very common combinations in the online reviewing system. We choose the "Health & Personal Care" category of Amazon review [7] as the data set we're going to build and test our model. The first thing we do is performing some analytical exploration on the data set in order to better understanding the properties of the data set and finding possibly useful features. Second, we build a simple model that consider only the global average and the offset of each word in the review texts as our baseline model. After that, we build three models, the first one is latent factor model that only considers the reviewers and products, the second model we add on the interaction between words and other features and the last one is the support vector regression model. In the end, we compare the performance of each model and draw a conclusion.

Keywords

latent factor model, linear regression, support vector regression, rating prediction, matrix factorization, SVD, data analysis

1. INTRODUCTION

The rating prediction has been playing a important role in nowadays business. By predicting precisely the rating for a user, the companies can get access to the answer of the following questions like 'What will he/she buy together with this product?' or 'Does he/she like this kind of product?'. With such data, the companies would be able to conduct some statistical analysis and make some business decisions like whether to cut the budget of that item . The same story happen to customers. They need the predictions of a product to help them decide whether they will buy it or not. In our paper, we try to make the best prediction on rating using reviewerID, productID, review text and review time. We compare the result of several different models like latent-factor model and SVM in the following sections and make conclusions at the end of the paper.

The rest of the paper is organized as follows: section 2 introduces some statistical analysis on our data set. Section 3 describes our prediction task and how we evaluate our work. Section 4 covers the whole process of our research, including set a baseline, optimizing different models, make comparison and some interesting findings. Section 5 shows the result of our experiment and makes conclusions on our work.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. DATA ANALYSIS

2.1 Data property

We decide to use the amazon review data [7] and focusing on "Health and personal care" category. The whole data set contains 2,982,356 reviews with 1,851,154 unique reviewers and 252,331 unique products. In average, each reviewer only gives 1.61 reviews and each product receives 11.82 reviews. Since the data set is very large, we randomly extract approximately 33% of the entire data set as the training data, which is shown in the Table 1.

In order to validate and test our model, we extract two other disjoint data sets as validation data and testing data. Each of the data set are about 13% of the whole data set. Cold start problem is an important issue for recommendation system. As a result, we calculate how many reviewers and products in the validation and testing data have never be seen in the training data. This is a measurement of how serious the cold start problem is. Table 2 shows the result. As we can see, nearly 75% of the reviewers and 30% of the products are new to the training data.

2.2 Statistical analysis

In this paragraph, we're going to do some statistical analysis on both training data and the entire data set first. Then we'll compare the result and draw a conclusion. We'll use Dt as the shorthand of training data and D as the entire data. Since it is a prediction task, the distribution of the rating is fairly important. For Dt and D, the average rating are both 4.108 to the third digit after the decimal point. The distributions are shown in Fig 1. The histograms indicate that over 75% of the reviewers rate the product higher or equal to 4.0.

Furthermore, we would like to find out whether there're some correlation between the rating and other features. First,

Table 1: overview of the data

	total review	# of unique user	# of unique item
whole data set	2982356	1851154	252331
training data	1000397	775622	157325
validation data	399220	347534	98613
testing data	400375	348297	98450

Table 2: new reviewer and product in validation and testing data

	total reviews	# of unique user	# of unique product	# of new user	# of new product
validation data	399220	347534	98613	261081	29524
testing data	400375	348297	98450	261533	29400

Figure 1: Distribution of each rating

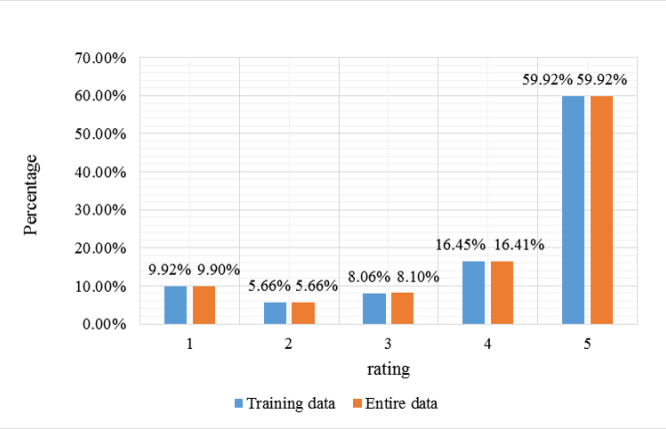
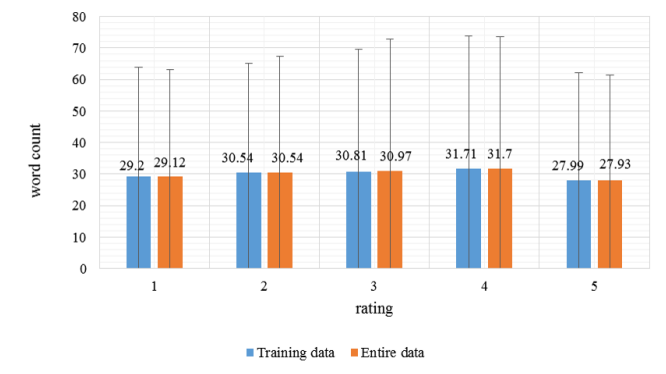


Figure 2: Average words in Review of each rating



we examine whether the number of words in a review text affects the rating. Fig 2 is the average number of words in Dt and D respectively. An interesting thing is that the average number of words has the smallest value when the rating is 5.0. However, the standard deviation of the number of words within the same rating group is so large that the average number of words seems meaningless.

Second, we want to find out the relation between time and rating. The time feature provided in the data is a string represent the unix review time. However, directly using the unix review time as a feature is not a good idea. We have to turn it into discrete, periodic time sequence like week or month which make more sense. We first calculate the

Figure 3: Average Rating on each day of a week

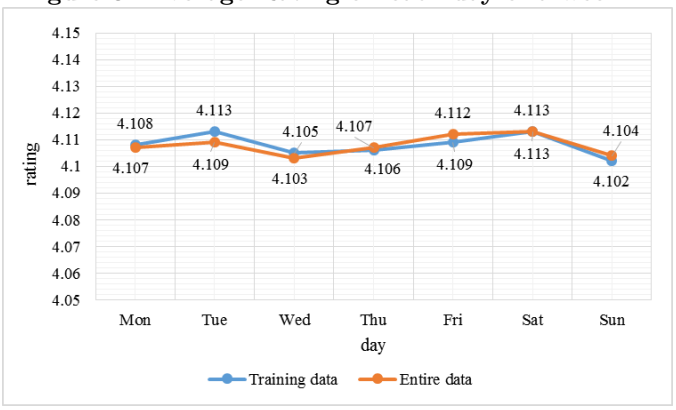
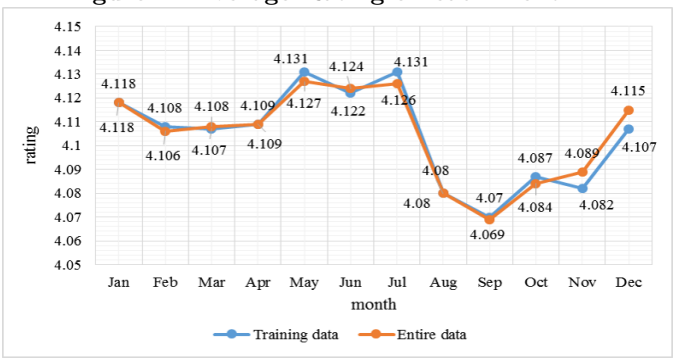


Figure 4: Average Rating on each month



average rating on each day of the week. Fig 3 shows the result on Dt and D.

As we can see, the rating is uniform in each day. So we consider there's no significant relation between weekday and rating. Then we calculate the average rating in monthly bases and we discover that from August through November, the average rating is lower than the other months. The results are shown in Fig 4.

To sum up this section, we find out that the distribution of our training data and entire data are very similar. So it shouldn't be a problem to use the training data set to train our model instead of using the entire data set. Second, though the length of the review text and the day of review seem to be irrelevant to the rating, the month of review suggests that the rating varies in different time of a year. This finding gives us a thought that perhaps the month of

review could be a useful feature.

3. PREDICTIVE TASK

Our main idea is to predict the rating based on reviewer ID, product ID, review text, and unix review time. As a result, we extract these features and eliminate others like helpfulness and summary. Ratings are discrete number scale from 1.0 to 5.0. reviewer ID and product ID are unique string for each user and product. Review text is a raw sequence of words written by the user. Unix review time is the time that this review was published.

Given a reviewer's review on an item, and going to predict the rating to product given by the user. Formally, let u denote the reviewer ID, i denote the product ID, r be the actual rating, T_{ui} be the review text given by the user and t_{ui} be the time that the review being recorded. We want to find $f(\cdot)$ such that:

$$arg \min_{f} \left(f(u, i, T_{ui}, t_{ui}) - r\right)^2 + \lambda(\Theta)^2 \quad (1)$$

We will use the data set described in previous section, and report RMSE on test set to evaluate our models.

4. MODEL

If T_{ui} and time stamp is not provided, it's a classic rating preiction problem. There is a classic latent factor model [6] for this kind of problem:

$$arg \min_{\alpha,\beta_u,\beta_i,\gamma_u,\gamma_i} \left(r - (\alpha + \beta_u + \beta_i + \gamma_u\gamma_i)\right)^2 + \lambda(\Theta)^2 \quad (2)$$

The model define the interaction between reviewer and product as inner product, and let the model find the embedded high dimension structure itself. There are many optimization methods for solving this kind of task, such as Stochastic Gradient Decent [6], Alternative Least Square [4], Monte Carlo Markov Chain [1] etc. We will base on this classic task, and discuss what we can do if review text is given as extra information.

4.1 Preprocess

In the previous section, we discover that the number of words in the review text can do nothing about the rating since the standard deviation is so large. It seems to be more reasonable to utilize the content of the review text. But before that, we do a little pre-processing about the review text. First, we turn all the words into lowercase and remove all the punctuations. After that, we remove the stop words based on the stop word list in nltk python library [2]. Finally, we use the Lancaster Stemmer provided by nltk python library to do stemming for each word. Another pre-processing we do is classify the unix time into each month. In the previous section, we find out that the average rating in August through November is much lower than other months. So instead of using the unix time directly, we classify them into twelve months.

4.2 Baseline model

The model of our baseline in this prediction task is:

$$rating = \alpha + \sum_{w \in T_{ui}} \beta_w / |T_{ui}| \quad (3)$$

First, we calculate the average rating for each word. Then we get the beta offset by subtracting the α, which is the global average rating. Finally, we sum the β, average rating, of every word in the review text and divide it by then length of the text, then add it to the global average rating. For example, intuitively, if the word 'fantastic' appears in the review text, there is a high possibility that the rating will be 4.0 or 5.0. Thus we go through all the review text and find what is the average rating of those reviews which contain the word 'fantastic'. If the result is 4.7 and the global average rating is 4.0 for example, it means that the word 'fantastic' has a positive impact on rating, at about 0.7. For some common word like 'the', because it almost appears in every review, so the average rating for 'the' will be just the same as global average rating, which means 'the' has no impact on rating, matching our common sense. To evaluate our baseline model, we compare it with the simplest model - just predicting the global average rating for all the reviews. The result of the experiment showed that the RMSE of the baseline on the test set is 1.26873, while the RMSE of the simplest model is 1.33312 Thus It turns out that our model is better than the simplest model, thus to be a valid baseline.

4.3 Model

4.3.1 Support Vector Regression

In many practice situations, linear model does great jobs if we manipulate features in a correct way. To capture the embedded high dimension structure of review text feature, we treat review text as a sparse binary vector weighted by reciprocal of length of review text:

$$\underbrace{(0, 0, 1/|T_{ui}|, 0, 1/|T_{ui}|, ..., 0, 1/|T_{ui}|, 0, ..., 1/|T_{ui}|, ..., 0)}_{|W|} \quad (4)$$

The 1 in the vector indicates that we have such word in review text. This treatment is useful in situations that we do not know the embedded relation between features and the label. Similar technique is used on reviewer ID, product ID and timestamp feature because there is no information in ID number and number of timestamp, so we expand it as a binary indicator vector. Especially, as the analysis result in section 2, we choose to use 'month' as our timestamp feature and expand it up to 12 dimesion feature. The whole map of feature looks like:

$$\begin{aligned}(1, \underbrace{0, 0, 0, ..., 1, ..., 0}_{12}, \underbrace{0, 0, 0, ..., 1, ..., 0}_{|U|}, \underbrace{0, 0, 0, ..., 1, ..., 0}_{|I|} \\ , \underbrace{0, 0, 1, 0, 1/|T_{ui}|, ..., 0, 1/|T_{ui}|, 0, ..., 1/|T_{ui}|, ..., 0)}_{|W|}\end{aligned} \quad (5)$$

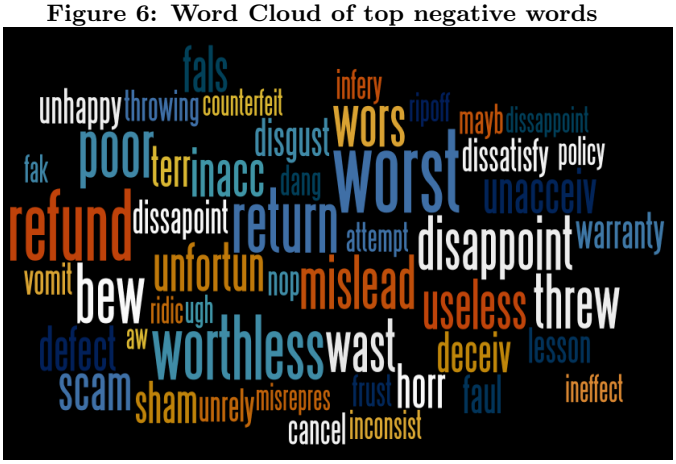
Though it's a very high dimension vector, it's extremely sparse. So, we choose to use *LibLinear* [3], a powerful learning tool for massive dataset, to handle this kind of feature. Another reason to choose linear model is that we can easily see which features are important and which are not to justify our approach.

4.3.2 Latent Factor Model

Based on the success of latent factor method which model the reviewer-product interaction as inner product, we choose to model the interaction between words and other features

Figure 5: Word Cloud of top positive words

Figure 6: Word Cloud of top negative words



in the similar way. Formally, we fit the model:

$$arg \min_{\alpha,\beta_u,\beta_i,\gamma_u,\gamma_i,\gamma_w,\gamma_{t_{ui}}} (r - (\alpha + \beta_u + \beta_i + \gamma_u\gamma_i + \gamma_u\gamma_{t_{ui}} + \gamma_{t_{ui}}\gamma_i + 1/|T_{ui}|(\sum_{w\in T_{ui}} \gamma_w\gamma_u + \sum_{w\in T_{ui}} \gamma_w\gamma_i + \sum_{w\in T_{ui}} \gamma_w\gamma_{t_{ui}}) + 1/|T_{ui}|^2 \sum_{w\in T_{ui}} \gamma_w^2)^2 + \lambda(\Theta)^2 \quad (6)$$

This model is similar to *SVD++* [5] with additional interaction. The idea behind is that we would like to know how reviewer writes such words in the review text, how such word on the product and how two words appear in the same text contribute to the rating. Similarly, we also add timestamp feature as interactive variables. Similar as the previous section, we consider 'month' as our timestamp feature. Though it looks complicate, we are still allowed to use similar techniques, such as SGD, for fitting classic latent factor model.

5. RESULTS AND DISCUSSION

We list our results in Table 3. Our results are much better than the original Matrix Factorization method even the simplest baseline model. One reason is that review text provides more information about the rating. Another reason is that because there are many new reviewer IDs and product IDs which are not in the training set, traditional Matrix Factorization method can do less things in this situation. However, review texts help us find the hidden high dimension structure of new reviewer or new item, mentioned in section 2.1 and 4.3.2, and resolve the cold start issue.

Moreover, as mentioned in section 4.3.1, we can take a look at the weights of each word contributes to the rating in our linear model. Figure 5 and Figure 6 show us which words in the review text have significant relation with rating. In Figure 5, there are some positive words, like "wow", "delight", "favorite",... on the other hand we can see negative words, like "worst", "worthless", "refund",... in Figure 6.

Besides, 'month' feature also enhanced the performance of our latent factor model which becomes our best overall. This result meet our expectation in the analysis section. However, timestamp feature did not improve the linear model. So, there is nonlinear relation between 'month' and rating which cannot be discovered by linear model.

In sum, we can say that review text and timestamp are powerful features to predict the rating that the reviewer gives to the product if we carefully design the models. In addition, we can rely on it to solve the cold start issue in many rating systems.

Table 3: Results

Model	RMSE
Global average	1.33312
Baseline model	1.26873
Support Vector Regression	1.02986
Support Vector Regression (with time info)	1.0303
Latent Factor Model (U x I)	1.30683
Latent Factor Model (U x I x W)	1.06557
Latent Factor Model (U x I x W x Time)	1.01505

6. REFERENCES

[1] C. Andrieu, N. de Freitas, A. Doucet, and M. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[2] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[4] H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method.

[5] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[6] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[7] J. J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. *CoRR*, abs/1506.08839, 2015.