Extracting Rating Dimensions from Hidden Topics in Text Reviews: a Better Recommendation System

Xuwen Shen
Department of Statistical Science
Cornell University
Ithaca, NY
`xs288@cornell.edu`

Xinzhe Yang
Department of Computer Science
Cornell University
Ithaca, NY
`xy269@cornell.edu`

Abstract

The ultimate goal for the project is to create a personalized recommendation system which recommends restaurants to a specific user according to the user's preference and the restaurants' rating with respect to the user's preference.

In order to give insights to overall ratings and then create a new personalized recommendation system based on the rating that account for his or her preference, we were hoping to extract hidden information in reviews including an individual user's preference and a business's properties (scores for each feature of the business). We used LDA to get topics from the review. If we have k topics in reviews then we extract k (the same number as the number of topics in Yelp reviews) dimensions in rating for each topic respectively and then use the k dimensional rating to compute the recommendation score for an individual. And then we created a model combining the topics and overall ratings to get a personalized ratings for a specific user. Finally the method is validated by comparing it with the baseline.

1 Introduction

The same rating from different users usually stands for different meanings. Individual users may assign different weights to such aspects when determining their overall score.For example, a spendthrift hotel reviewer might assign a low weight to "price" but a high weight to "service", thus explaining why their overall rating differs from a miserly reviewer who is only interested in price. There exists hidden information in reviews that leads to the final rating. By extracting the information, we can get where a restaurant shines and what it needs to improve. If we have k topics in reviews then we extract k (the same number as the number of topics in Yelp reviews) dimensions in rating for each topic respectively and then use the k dimensional rating to compute the recommendation score for an individual.

The importance of customer reviews has been proven in terms of giving insights to businesses and also providing customers personalized services.

However, few of previous researches have been studying both the topic models of a specific user and those of a specific restaurant with rating break-downs.

Currently, Yelp has a recommendation that does not take the user's preference and potential matching with restaurants into account. Our recommendation will benefit customers by providing a more personalized user experience.

2 Background

Prediction methods based on user's previous rating for items or business are commonly used to build a recommendation system. To make improvements on prediction results and take a good advantage of text reviews, we were hoping to extract hidden information from text and make predictions along with overall rating. More importantly, we want to personalize the recommendation by predicting a rating for one business from one specific user.

Here we introduce two related work by previous scholars. The most similar line of work is "Hidden Factors of Topic" model[4]. Hidden factors here refer to hidden rating dimensions related to contents in hidden reviews. The model used a transformation function to link the topic probability and a business's properties (scores for each feature of the business). The "Hidden Factors of Topic" model facilitate tasks of linking the topics in reviews with overall ratings. Here's another model, "Codeword"[3], which makes it easier to extract sentimental information from text. It gives us a positive or negative description for the topics which we want to extract from text by adding specific words to text. The two methods inspired us and led us to our own method below.

3 Research Design

Our objective is to learn hidden dimensions of behind a overall rating for a specific restaurant by combining latent rating dimensions, such as the topic learned by topic modeling methods like LDA. First we run topic model to get topics from reviews written by a user and topics from reviews on a restaurant. Then we combine topics we learned from the user and the restaurants as topic factors. Notice that the user preference and topics of restaurants is the same number, as well as topic factors. Finally, we learn the hidden dimension of rating by minimizing the mean squared error function defined by difference between the overall rating calculated by scores of hidden dimensions and the real rating of a review from one user for a restaurant.

In order to do that, we collected all the reviews from one user to get topics from the text as well as the probability of the topics. In effect, the probability of the topic in all the reviews from each user can be the "preference" of the user. After we got the topics of the user, we subsequently used the same model on all the reviews for one specific restaurant. However, the results we get from LDA lack the adjectives that can distinct the positive or negative quality of the topic. So our next goal is to learn the positive or negative quality of topics.

After achieving the goal of finding the distinct quality of topics, we can then give a more reasonable initial values of rating for one feature of a business and train the values by minimizing the loss function.

We begin briefly with the standard model for recommendation model and Latent Dirichlet Allocation.

3.1 Standard recommendation model

The "standard" latent-factor model predicts ratings $r_{u,i}$ for a user u and item i according to $r_{u,i} = \alpha + r_u * r_i$ where α is an offset parameter, u and i are user and item biases, and u and i are K-dimensional user and item factors (respectively). Intuitively,i can be thought of as the "properties" of the product i, while u can be thought of as a user's "preferences" towards those properties. Given a training corpus of ratings τ, the parameters $\Theta = argmin 1/\tau \sum (rec(u,i) r_{u,i})^2 + \lambda\Omega(\Theta)$ are typically chosen so as to minimize the Mean Squared Error (MSE).

3.2 Latent Dirichlet Allocation

Each topic k also has an associated word distribution, ϕ_k, which encodes the probability that a particular word is used for that topic. Finally, the topic distributions themselves (θ_d) are assumed to be drawn from a Dirichlet distribution. The final model includes word distributions for each topic k, topic distributions for each document ϕ_k, and topic assignments for each word $z_{d,j}$. Parameters $\Phi = \theta$, ϕ and topic assignments z are traditionally updated via sampling. The likelihood of a particular text corpus τ (given the word distribution and topic assignments for each word) is then[1]

$$p(\tau|\theta, \phi, z) = \prod_{i} \prod_{j=1}^{N_d} \theta_{z_{d,j}} \phi_{z_{d,j}, w_{d,j}}$$

4 Methodology

We are going to use topic modeling methods analyzing text reviews and predict rating scores. Meanwhile, to verify that the impact of results comes from looking at text reviews, we also use a baseline method that does not look at text reviews at all, but merely uses collaborative filtering.

4.1 Baseline: Collaborative Filtering

4.1.1 Introduction to Collaborative Filtering

We used the method which builds a recommendation system without using text reviews as a baseline to give a efficient evaluation for our method. The baseline method we use here is Collaborative Filtering, which makes automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).

The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

This approach calculates the similarity between two users or items, and produces a prediction for the user by taking the weighted average of all the ratings. For example, the value of ratings user 'u' gives to item 'i' is calculated as an aggregation of some similar users' rating of the item: $r_{u,i} = \text{aggr } r_{u',i}$

Similarity computation between items or users is an important part of this approach. Multiple measures, such as Pearson correlation and vector cosine based similarity are used for this. In this paper, cosine similarity is used to calculate the similarity between items or users.

4.1.2 Method

To conduct the experiment, we used two datasets: user and business on Yelp to get a matrix containing the ratings for a business given by a user. Then we use the cosine similarity to get compute the similarity between users or businesses and thus we can decide "similar users" and "similar items". The unknown rating for a business given by a user can be decided as an aggregation of some similar users' rating.

4.1.3 Data

We used the the column of restaurant ID in business dataset, the column of overall rating in review dataset and the column of user Id in user dataset to create a matrix in which the entry is the rating for a business from a specific user with the row of user and the column of business.

4.1.4 Experiment

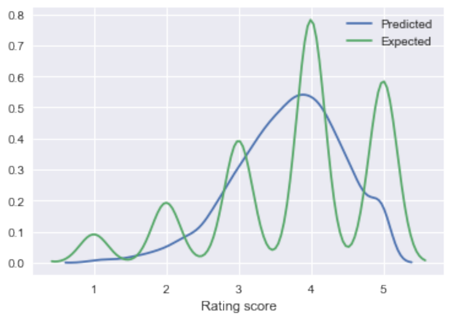
We chose all the business with the word "Restaurant" and "Food" as restaurants and selected from Madison. Before training, we divided our data into training set, which contains 70 percent of the whole dataset and validation set, which contains 30 percent of the whole dataset. Then we used cosine similarity to generate the similarity metrics. We used both "User based" and "Item based" Collaborative Filtering with KNN to predict rating for a restaurant from a user. For each method we tried 5, 10, 20 the total number of nearest neighbors respectively.

4.1.5 Results

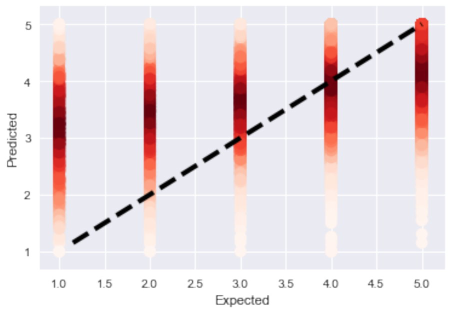
We can see from the table below: both methods with different numbers of nearest neighbor generate similar MSE and MAE with an average value of 0.86 and 1.40. The result table is shown in Table 1. Distribution plots are shown in Figure 1.

Methods	MAE	MSE
User-based: 5 nearest neighbors	0.872	1.256
User-based: 10 nearest neighbors	0.842	1.182
User-based: 20 nearest neighbors	0.832	1.159
User-based: All nearest neighbors	0.849	1.164
Item-based: 5 nearest neighbors	0.917	1.479
Item-based: 10 nearest neighbors	0.901	1.428
Item-based: 20 nearest neighbors	0.896	1.414
Item-based: All nearest neighbors	0.852	1.172

Table 1: Collaborative Filtering Results



(a) Gaussian kernal-density estimate



(b) Weighted Heatmap

Figure 1: Density distribution of Collaborative Filtering results

4.2 Latent Dirichlet Allocation

4.2.1 Introduction

The $n \times p$ matrix W in the above figure represents the rating score of each topic for each restaurant. Each row represents a restaurant and in that row, each column represents the rating score of a specific topic.

The $p \times 1$ matrix M on the right represents the user's preference for each topic. Multiplying the two matrices gives the personalized score of each restaurant for each user.

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{np} \end{pmatrix} \begin{pmatrix} u_1 & u_2 & u_3 & \dots & u_n \end{pmatrix}$$

4.2.2 Data

We use the datasets of yelp reviews, business and user to extract topics from reviews for popular restaurants (restaurants with most reviews) active users (users with most reviews). We picked all the restaurants in Madison in business dataset and we use the data with pre-known rating for a restaurant given by a user as samples. We split the data into training part and validation part.

For the text reviews, we preprocess them by tokenizing, extracting only nouns and adjectives, removing stop words, stemmizing.

After preprocessing, we turn our tokenized documents into a id to term dictionary and then a document-term matrix.

4.2.3 Method

Recall that the goal of building a recommendation system is to optimize the parameters in the function. Typically we initialize the ratings matrix randomly and then use the Gradient Descent to minimize the loss function. In our method, we were hoping to combine the results from text reviews when minimizing the loss function. Before this, we use the Latent Dirichlet Allocation to extract topics in all reviews first and generate k topics. Then for each user, we extract all the reviews as a corpus and calculate the probability that each word belongs to each of the k topics. Finally we add up and normalize to get the preference column of each user where there is a 0.0-1.0 weight for each of the k topics. This is based on the assumption that the user preference for any topic is correlated with the topical probability of his text reviews.

Then to find the rating scores for subtopics for each restaurant in the matrix W, we use the following formula to train a model to predict the score of subtopics. We minimizing the L1 loss of recommended score minus real rating score, using Sequential Least SQuares Programming provided by SciPy [2].

$$L = \sum_{i=i}^{n} |W_i \cdot M_j - star(i,j)|$$

where we are trying to find out W_i, a column of rating subscores for each of the k topics for some restaurant i, given M_j, a column of user j's preference for each of the k topics, and $star(i,j)$, the actual rating score we found in the dataset.

4.2.4 Experiment

To see the performance of the recommendation system, we chose to analyze the restaurants in Madison (**1231** in total). We split the preprocessed review dataset into 70% training set (**41926** pieces of reviews by **15728** users) and 30% testing set (**18758** pieces of reviews). Since the restaurants' topical subscores in the testing set have to be available, we performed the split among restaurants. In addition, we are predicting rating scores only if the restaurant had at least 10 available reviews to train topical subscores and the preference for that specific user was available from the training set.

We use mean squared error and mean absolute error to determine the overall performance of the algorithm.

For LDA algorithm, after trying out different combinations of parameters, we don't see much improvement in terms of prediction accuracy as shown in Table 2. In Table 2, k represents the number of topics and passes represents the number of passes through the entire corpus.

Combinations of parameters for LDA	MAE	MSE
k=20, $passes$ =1	0.877	1.404
k=20, $passes$=20	0.858	1.357
k=20, $passes$=40	0.857	1.353
k=60, $passes$ =20	0.855	1.356
k=120, $passes$ =20	0.865	1.354

Table 2: Results on Different Parameters for LDA

Choosing k=20 and $passes$ = 40, after running LDA on the test dataset, we approximated the preference column for each user on each of the k topics and the distribution of preferences for all users is shown in the Figure 2.

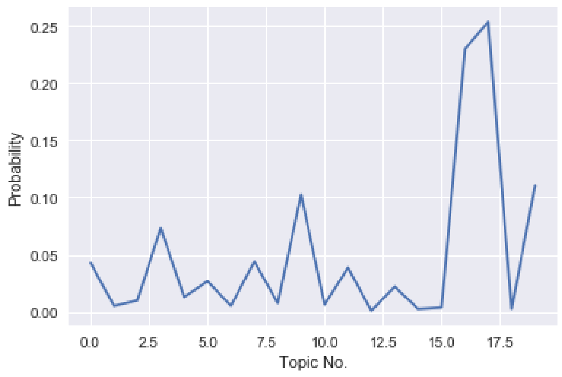


Figure 2: Preference distribution of users

From our observation, topics with high frequency tend to include general words instead of words talking about any specific food. For instance, in Table 3, the top 10 words produced on topic No.1 are clearly talking about breakfast or brunch, those on topic No.14 are talking about Japanese food. On the other hand, topic No.16 and topic No.17 include only general features about restaurants instead of specific food. This observation is consistent with logic behind how we calculated the topical preferences for each user.

Topic No. and probability p	Top 10 words from the topic				
No.1, $p = 0.005$	breakfast	brunch	toast	egg	bloodi
	french	morn	mari	bacon	sunday
No.14, $p = 0.002$	sushi	roll	tuna	madison	salmon
	fish	chef	Japanes	fresh	tempura
No.16, $p = 0.230$	coffe	place	good	cafe	shop
	nice	cup	great	bakeri	littl
No.17, $p = 0.254$	food	time	servic	order	tabl
	server	restaur	waitress	busi	experi

Table 3: Top 10 words produced on specific topics by LDA

For the above results, we are predicting rating scores only if the restaurant had at least 10 available reviews to train topical subscores. However, if we change this number, the performance of the recommendation system will change accordingly. More specifically, the performance is positively correlated with number of text reviews available to be trained by the restaurants as shown in Figure 3. This observation is indeed reasonable in that we used text reviews as the major component to train restaurants' topical subscores and therefore text reviews contribute a large part to the performance of our method.

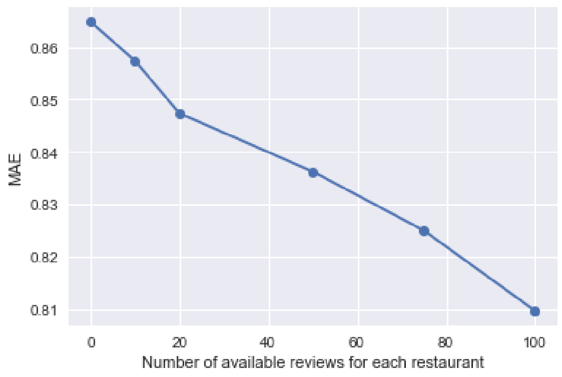


Figure 3: Correlation between performance and number of available training reviews

4.2.5 Results

Taking a deeper look into the results of our method, with k=20, *passes*=40 and predicting on restaurants that have at least 10 reviews to be trained, we get a result of **MAE=0.857** and **MSE=1.352**. In terms of error, our method is doing a reasonably good job and we are optimistic that the performance will improve with larger training data size, as shown in Figure 3.

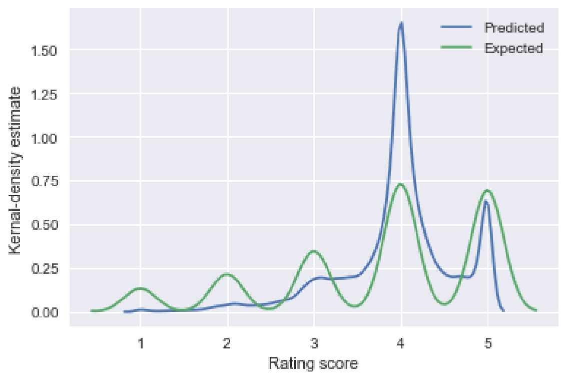
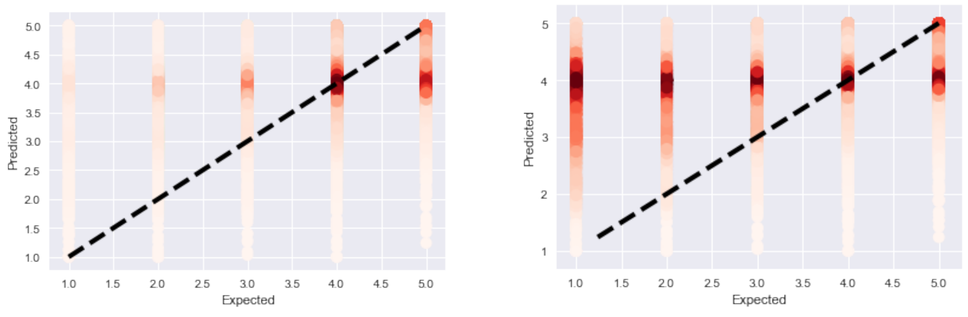


Figure 4: Distribution of predicted and expected ratings scores

However, one big issue with our method is the distribution of predictions. This happens because the Yelp dataset is a biased sample in terms of rating scores. The training dataset has 7.9% "1"'s, 9.9% "2"'s, 14.9% "3"'s, 31.4% "4"'s and 35.7% "5"'s, with a significantly portion of positive reviews. Therefore, since we are using L1 loss to train restaurant topical subscores, as shown in Figure 4, it's tempting for our method to predict positive scores, which results in a good mean absolute error.



(a) Unweighted (b) Weighted

Figure 5: Density distribution of predicted vs. expected scores

Figure 5 above shows the heatmap density distribution of predicted and expected scores and clearly our method is predicting "4" too frequently. There exists a trending in negative reviews that some of negative scores were predicted.

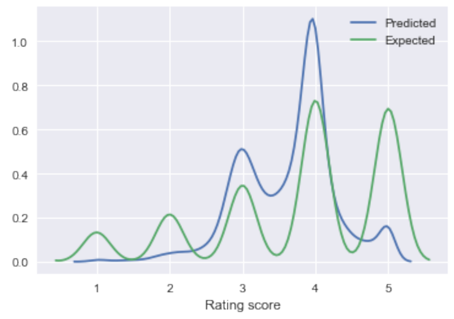
Overall, our method does a reasonably good job on predicting scores in terms of MAE. However, it tends to predict negative reviews as positive ones due the biased training sample.

5 Further Analysis

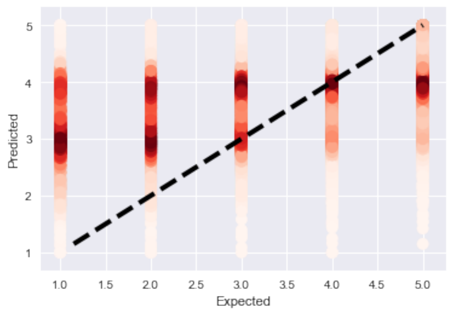
As discussed earlier, our method is not doing a good job at predicting negative values, which is a classic problem due to the biased feature of Yelp dataset. We tried to use L2 loss to train restaurant topical scores instead, but neither MAE/MSE nor distribution significantly improved.

Therefore, we propose that modify the loss function so that predicting negative reviews as positive reviews results in more penalty. Tentatively, we set square the error if the predicted is 2.0 larger. The resulted MAE and MSE are 0.904 and 1.326 respectively, and plots are shown in Figure 6. The errors are larger but it's slightly better at predicting negative reviews. However, this is still not an ideal improvement yet. We can still do a lot to resolve this issue by investigating into a better choice of loss, regularization, bias functions.

An alternative proposal is to perform over-sampling on negative reviews in the training dataset, that is, to generate new samples in the scores which are under-represented. Once we have a more balanced training sample, the prediction results are expected to improve.



(a) Gaussian kernal-density estimate



(b) Weighted heatmap

Figure 6: Performance of proposed improvement

6 Conclusion

On hypothesis testing the MAE results for methods on 5 cities, we have p-value equal to 0.4988, which is greater than 0.05, so we do not reject the null hypothesis that true difference in means is equal to 0, that is, in terms of MAE, both methods have similar performance.

Given the evaluation results measured by MAE and distribution of predicted value and expected value for both baseline and our method, we can conclude that our method is slightly better than the baseline in spite of similar MAE. On the one hand, Figure 4 shows that in our method, the predicted value catches the curve better while the the baseline method fails to catch the trend of rating of 5. On the other hand, from Figure 3, MAE has significantly decreased as the number of available training reviews increases, which strongly indicates that we can get better results with a larger number of available text reviews. Given the fact that we only used a small part of text reviews to train our model, we deduce that our method can perform significantly better than the baseline.

Moreover, both Figure 2 and 3 provide sufficient evidence that the impact of results of our method comes from looking at text reviews, which Collaborative Filtering does not.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[3] Jack Linshi. Personalizing yelp star ratings: A semantic topic modeling approach. *Yale University*, 2014.

[4] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.