Semantic Feature Analysis and Mining for Yelp Rating Prediction

[CSE255 Assignment1]

Yinshi Zhang
Computer Science and
Engineering, UCSD
yiz205@ucsd.edu

ABSTRACT

In this project, we investigate proper features of Yelp data to build machine learning models for rating prediction and recommendation tasks. Firstly, we described a linear regression model to observe contribution and influence of various features from data of Users, Business, Checkin and Reviews to predict review rating. Particularly, we discovered that information from review text is very useful for understanding rating scores, however which is often ignored by traditional prediction approach. Then, we applied discoveries from feature analysis to improve traditional latent-factor recommendation model by integrating implicate linear features as side sources. Further, we extracted textual topics by Latent Dirichlet Allocation (LDA) algorithm and incorporated them as latent factors of users or business to recommendation model. The recommendation model with textual information achieved Root-mean-square-error(RMSE) of rating prediction as 0.9591, improved 14.28% compared to baseline.

Keywords

Feature Engineering, Recommendation System, Data Mining, Topic Mining, Yelp Challenge

1. INTRODUCTION

Features encode information from raw data that allows machine learning algorithms to classify an unknown object or predict an unknown value. Feature engineering is one of the most important factors in a machine learning project. The goodness of feature selection is vital to success of learning model. Thus in this project we want to implement a feature analysis to understand the properties of the data and task to solve and how they might interact with the strengths and limitations of the classifier for advanced predictive tasks.

In this project, our motivation is from recommendation system design. Recommendation system is used to recommend products to users by predicting user's rating to candidate items [1]. In order to build a model to predict user's evaluation to an unknown item, it is vital to reveal properties of users and products which effect the final evaluation significantly. For example, obviously average of user's former rating reflects how critical the user is and review count of a product may reveal the popularity of the product, which are both important information to predict rating. Moreover, we may assume user's age, living place, as well as products special properties may also influence the evaluation. However, due to scale of data collection, extremely high dimensional feature set may cause unexpected noises or high complexity of computation. So we intend to discover features fit into prediction model well and informative to recommendation task.

Traditional recommendation approaches are mainly in two categories, memory based collaborative filtering and latent factor model based collaborative filtering [8]. Memory based collaborative filtering tends to use statistical techniques to find similar users to the user being predicted, then based on behavior of the candidate users to make a prediction to a particular user's evaluation on the item. Latent factor model based collaborative filtering approaches utilize the dataset to learn a probabilistic model to represent features of users and items as well as their relationship. Latent factor model is proved achieving higher performance than memory based approach [5], however lack of semantic interpret of the model since latent factors has no side feature associated. Besides, cold start problem is challenging for accuracy to predict new users behavior with both approaches. That means besides rating information, other feedback is needed. Thus, it is necessary to incorporate implicate features other than rating to the model.

Furthermore, with development of various review websites, plentiful functions on websites provides more meaningful feature besides pure numerical rating. Obviously, users textual reviews are majority portion of data from review website. Understanding and learning content of reviews can provide deeper insight to hidden information of the rating. For example, a user gives lower rating than average because he is strict to service while in fact he doesn't care about food flavor much, but the bias of this user in latent factor model can only reveal the overall rating habit of the user. However, this user must have asserted his reasons for low rating in reviews or comments. On the other hand, side features of

businesses can also help them aware strength and shortcoming of themselves and user's preferences so than improve in a right direction. Hence, inspired by McAuley's HFT model [14] but due to time limit, we intend to implement a simpler appoarch by learning topic model from reviews text with LDA, then incorporate probability vectors into recommendation model as hidden factors.

Our main contribution is to analyze informative features to improve accuracy of rating prediction(recommendation). Firstly, we present that the features we selected can reflect rating well and abandon noisy features. Secondly, we prove high contribution of textual information to prediction. Further, we extract topics to reveal deeper insights of Yelp data. Finally, combining semantic features with rating based recommendation model can perform lower error rate by 14.28%.

2. RELATED WORK

With development of e-commerce and review websites, recommendation system plays a vital role to help customers search and make decision easily from large scale information. As a result, recommendation system design is widely researched towards a variety of business areas, such as movie review from Netflix [3], Amazon datasets [12], beer and wine reviews [15], news recommendation [16].

Latent topic modeling is very widely used as an unsupervised model for clustering and classification in both natual language processing, computer vision and other machine learning areas. LDA is a common method of unsupervised learning to discover hidden topics. It assumes that there are latent variables that reflect the thematic structure of the documents [4]. Another common topic modeling method is probabilistic latent semantic analysis (PLSI) which is not a strict generative model [7].

As mentioned in Introduction section, HFT [14] combines latent rating dimensions and latent review text dimensions, which results in more interpretable topics and more accurate rating predictions at the same time. Besides, there are several related works discuss relationship of text content and rating stars for review website One related study, for example, uses a traditional LDA to discover hidden topics [9], and then uses these hidden topics to predict star ratings by averaging the star ratings of all reviews for businesses that contained a particular topic. This work concentrates more on discovering subtopics for a business instead of rating prediction for recommendation. Another related work tend to incorporate rating into LDA model as a parameter to generate better topics with sentimental meaning and easy to understand by human. Another study intends to improve recommendation accuracy and rating prediction accuracy by grouping users together with clustering techniques based on review topic [10].

Compared to HFT, our work is not limited to textual feature but general feature engineering for recommendation tasks. Although our approach as simplified version is lack of interactive factors between text and rating, it is much straightforward and adjustable for future work. Integration of latent factors and linear semantic features for prediction is similar to classifier ensembles. In fact, side feedback vector is learned as regression task. That is one of reasons why we can discover good features simply by linear regression model. Classifier ensembles are known to offer a significant improvement in prediction accuracy. Ensembles include changing the instances used for training through techniques such as Bagging [2] and Boosting [6].

3. DATASET STATISTICS

We use data from Yelp challenge dataset which includes data of businesses, business attributes, check-in sets users and reviews. The details about the dataset are available on the Yelp website.[1] In this section we overview the key properties of the data set that we use for feature generation and modeling. We separate 20% of the whole dataset as test data. Table 1 shows general information of the dataset.

	Training set	Test set
Number of users	225,926	99,338
Number of business	41,745	32,272
Number of review	896,099	226,868

Table 1: Training and test set properties

3.1 Properties

The review table includes information about each review. Specifically, it contains business_id, user_id, stars(A star rating on a scale of 1-5), text (The raw review text), data, votes(The number of 'useful', 'funny' or 'cool'). The user table consists of user_id, name, review_count, average_stars (Average rating on a scale of 1-5 made by the user), votes(the total number of votes for reviews made by this user). business table contains details about business including id, name, neighborhoods, address and geographic information, stars, reviews_count (The total number of reviews about this business), categories (a list of category tags for this business), and other attributes. Checkin data is associated with business, containing number of checkins for different time periods in a day.

3.2 Statistics

We first examine whether Yelp graph follows a power law distribution. Figure1 illustrate the degree distribution of each node type in a log-log scale(user-review count and business-review count). We can see that that businesses are distributed according to the power law, while users are even at the log scale the distribution follows an exponential pattern. This phenomenon confirms our assumption of cold start problem of traditional latent factor recommendation model, and even proof that similarity based model is useless on Yelp data. Thus, we are convinced that feature analysis serves as a vital role in prediction and recommendation tasks.

Then, we try to figure out distribution of review rating. Surprisingly, the review distributions in our subset were skewed to the 4 and 5 star categories heavily. They consist of around 80% of the distribution, whereas the 1, 2, and 3 star categories are each only around 10-15% at most. Further, we plot relationship of average rating with review_count, as shown in Figure 3 There is a slight decreasing trend with increase of number of business the user rated. It follows the common sense that experts are more critical to items.

[1] `http://www.yelp.com/dataset_challenge/`

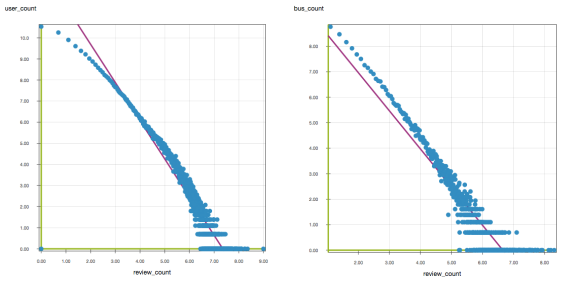

Figure 1: Left: Degree Distribution of User Review Counts; Right: Degree Distribution of Business Review Counts

Value	Count	Percent
5	406,045	36.078%
4	342,143	30.4%
3	163,761	14.551%
1	110,772	9.842%
2	102,737	9.128%

Figure 2: Rating Stars distribution

Another interesting function of Yelp provides is to vote a review. We make a simple scatter graph with number of votes of reviews get in three types, 'useful', 'funny', and 'cool'. Interestingly, as shown in Figure 4, most highly voted reviews as 'useful' and 'funny' are tend to be negative, while 'cool' is likely to be voted to positive reviews.

Although Yelp Challenge data set collected data from Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison, we find out that it is imbalanced on location. Figure 5 shows number of reviews from each city (We only show the top 15 due to limit of space). We can see most data is from Las Vegas. We assume that geographic information of the full scale data maybe useless for prediction since feature as a vector of city names is too complicated and noisy, which we can see from result in next section. However, we believe geographic information can be used within in specific location range.

Categories tags of a business is added by owner, so there is no uniform format and count, which makes it hard to extract them as feature. We tried to pre-process categories tags by converting list of strings into "bag of words". For train data

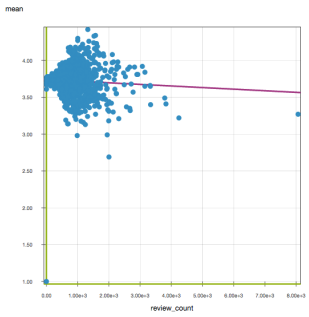

Figure 3: User Average Rating vs Review Count

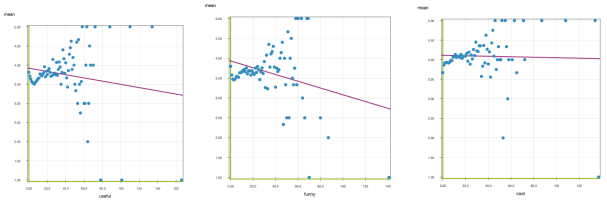

Figure 4: Average Rating vs Votes Count. Left: useful, Middle: funny, Right: cool

Value	Count	Percent
Las Vegas	417,763	46.416%
Phoenix	148,278	16.475%
Scottsdale	82,259	9.14%
Tempe	44,039	4.893%
Henderson	32,104	3.567%
Chandler	27,018	3.002%
Madison	25,033	2.781%
Mesa	24,984	2.776%
Edinburgh	17,050	1.894%
Gilbert	16,604	1.845%
Glendale	15,332	1.703%
Peoria	7,404	0.823%
North Las Vegas	6,329	0.703%
Surprise	4,633	0.515%
Goodyear	3,966	0.441%

Figure 5: Businesses of cities distribution

set, we get 738 unique tags. Intuitively we assume genre or cluster information might be useful for rating prediction, especially recommendation task. For example, most people have particular preferences towards one genre of music or movies.

4. FEATURE ANALYSIS FOR RATING PREDICTION

In this section, we intend to investigate how each feature of users, businesses, reviews and checkins influences rating stars of a user-business pair. For simplicity, we start with linear regression model.

4.1 Algorithm of Linear Regression

Linear regression models the target Y as a linear funcion of the feature variables X_j, a bias term (α) and regulation term λ:

$$Y = \alpha + \sum_i w_i X_i + \lambda \tag{1}$$

The coefficients (w_i) are what the training procedure learns. Each model coefficient describes the expected weight of influence in the target variable associated with feature. Intuitively, the coefficients often tell an interesting story of how much each feature matters in predicting target values. the bias term indicates the average target value. For example, in the business rating on Yelp, the value of coefficient shows strength of the feature and the sign of coefficient (positive or negative) indicate direction of association to final rating.

4.2 Experiment

Here, we experimented on different feature sets for linear regression model and compare performance of rating predic-

feature	train RMSE	test RMSE
Baseline(Overal Average)	-	1.2993
AvgRating	1.0282	1.0258
AvgRating + review_count	1.0281	1.0257
AvgRating + review_count + city	1.0279	1.0256
AvgRating + review_count + date	1.0257	1.0269
AvgRating + review_count + checkin	1.0281	1.0279
AvgRating + review_count + fans	1.025	1.0256
AvgRating + review_count + votes	1.0076	1.0065
AvgRating + review_count + categories	1.027	1.0254
AvgRating + review_count + text	0.9722	0.9702
AvgRating + review_count + votes + text	0.9568	0.9557

Table 2: RMSE of Linear Regression with different feature set on rating prediction

tion by root square error(RMSE). Overall result is shown in Table2. We use average rating of all review as baseline.

Intuitively, average rating of users and average rating business got previously inflect rating habit of a user and performance of a business well. We use these two features to train the linear regression model. In addiction, as statical result of showed above, count of reviews may influence rating. However, the result shows that weight of review_count is very small compared to average rating(Table 3). Also RMSE shows little difference between model with and without review_count. However, review count is not a high dimensional feature, we keep it in feature set of further experiments. Predict from each star rating level is shown in table 4. It shows the model does well on ratings that were between 3 and 5 but not too well on ratings 1 and 2. One reason why this could happen is that the number of 4 and 5 star reviews is more than twice the number of reviews with 1-3 stars and the model prediction tend to have less deviation than real rating.

name	coefficient
(Bias)	-2.13835494271
user_avg_stars	0.810139715818
business_avg_stars	0.758439606128
user_review_count	3.61385994753e-05
business_review_count	1.65051253198e-05

Table 3: Coefficient Trained

As we discovered in overall statistics part, votes have special meaning to rating. Here, we add vote feature on model with average stars and review count. From the result, we are surprised to see a large decreasing on error. The result confirm our estimation of votes contribution, so we would add it into recommendation model.

Actual-Rating	Count	Avg of Predicted-Rating
1	22229	2.60742557405
2	20602	3.23249814299
3	32979	3.50614638428
4	68613	3.78243566511
5	80997	4.22787416076

Table 4: Avg of Predicted-Rating

Since, review text data is the most informative data for customers of review website, in order to take advantage of text data from a text mining perspective, we apply simple text processing to the raw text of Yelp business reviews to extract features so that improve our linear model's predictions. Our belief is that negative words such as 'awful' and 'disgusting' are useful in predicting when a rating will be bad. In addition, there are typical 'good' words and neutral words in existing natural language dataset like WordNet. For simplification, we define a tag dictionary containing 10 negative words, 10 positive words and 10 neutral words then construct a feature that captures these words in form of "bag of words".

Negative: 'hate','awful', 'disgusting','rude', 'dirty', 'slow', 'angry', 'disappointed', 'wait', 'horrible';
Positive: 'awesome', 'good', 'like', 'recommend', 'nice', 'discount', 'love', 'best', 'healthy', 'great';
Neutral: 'Chinese', 'Asian', 'Mexican', 'American', 'service', 'waiter', 'hour', 'morning', 'dinner'

As expected, performance for rating prediction improved significantly, where we can conclude that although our simple experiment only consider few seed words without complicated model justification, this type of feature engineering can make the difference for model goodness.

The performance of model with categories tags added doesn't improve much. One possible reason is that categories tags in Yelp are too noisy to fit into predictive model. Also, high dimension may cause difficulty in optimization. So we decide to discard this feature for recommendation model. However, as common sense, users' rating on different item genre can reflect users' preference. Latent factor model is able to capture that by latent factors but it would fail for "cold start" problem because the latent factor cannot be learned precisely by few data. In order to take advantage of genre feature and define a feature function to translate latent properties into feature vector that can fit into recommendation model, also considering good result performed by text feature, we propose to extract topics from review text.

Besides, other features like city, checkin are experimented but the results achieve little improvement. Basically the reason is that data is noise and dimension is too high on these features.

5. TOPIC MODELING

5.1 LDA model

For better explanation to motivation and results, we give a brief description of topic modeling algorithm, LDA. Latent Dirichlet Allocation (LDA) [4] is a Bayesian generative model for text. It is used as a topic model to discover the underlying topics from text documents. LDA assumes that a corpus of text documents cover a collection of K topics. Each topic is defined as a multinomial distribution over a word dictionary drawn from a Dirichlet $\phi_k \sim Dirichlet(\beta)$. Each document from this corpus is treated as a bag of words of a certain size, and is assumed to be generated by first picking a topic multinomial distribution for the document $\theta_d \sim Dirichlet(\alpha)$. Then each word is assigned a topic via

the distribution θ_d, and then from that topic k, a word is sampled from the distribution ϕ_k. θ_d for each document can be thought of as a percentage breakdown of the topics covered by the document.

5.2 Topic Result

In our experiment, we simply use Topic Modeling Toolkit [**?**] based on LDA algorithm provided by Dato Create python library, which use Gibbs Sampling as learning algorithm for LDA. We perform a cross validation to chose optimal number of topics as 30 and number of iteration as 30.

As an example, we show topic 10 words (ordered by distribution) for 5 topics on a 30 topic LDA model in table 5. Service, for example, is made up of words such as "service", "table", and "server". Latent topics can also contribute to other interesting discovery. For example, some temporal topics arise, such as the breakfast, lunch, and dinner categories. These will provide rating information related to time phrase when compare them to check-in sets.

Service	Food	Breakfast	Hotel	Club
service	good	breakfast	room	pretty
table	taste	coffee	crust	vegas
server	ordered	eggs	free	times
minutes	flavor	bacon	expensive	nice
asked	saucez	morning	stay	friends
manager	small	pancakes	vegas	strip
waitress	bland	french	pizzas	play
seated	pork	toast	casino	drinks
arrived	full	brunch	style	Club
check	half	bagel	small	nights

Table 5: Topic 10 words for 5 Example topics from Reviews for Restaurants (K=30): Topic labels are manually added

5.3 Interesting Discoveries

Due to the time limitation and low performance of device, we are only able to experiment on subset of Yelp data. Here, we select Reviews only for 'Restaurants'. The partial dataset contains 11,537 businesses, 43,873 users, and 229,907 reviews.

5.3.1 *Dimension of Review Rating*

With topic model, we can predict topics for each document with output as a list of K topics along with probability that document belongs to topic K. This leads to natural interpretations of rating dimensions, for instance we can automatically discover that a restaurant ratings are divided along topics like service, time phrase or genres of food. We randomly select a review as an example to show this insight benefit.

"Good choice for an affordable sushi dinner. Portions and quality are good. Prices (including sake) are reasonable. When I travel to Phoenix on business, this is on the list to visit."

Top 5 subtopic with high probability for this short review is shown in figure 6.

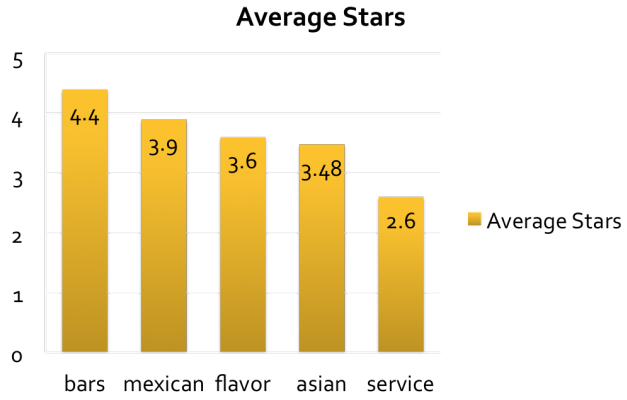

Figure 6: Example subtopics of review with average rating

5.3.2 *User Attention*

To get latent feature about a user, we take a user with relatively high degreee (number of reviews: 779) as an example.

According to result shown in figure 7, the user cares the most about Mexican food of all of these subtopics, making up 38% of all reviews. Users also care greatly about flavor of food, Asian food and sometimes visits steak or chain restaurant.

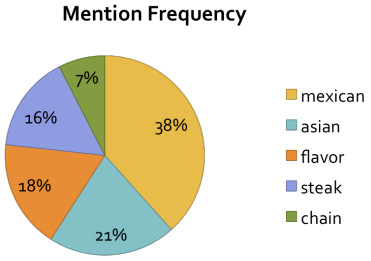

Figure 7: Mention Frequency of User Reviews

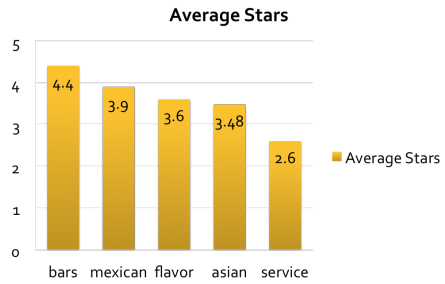

Figure 8: User's Average Rating on topics

5.3.3 *User rating criterion*

To recommend items for user, it is better to understand user's rating criterion. In common sense, people hold different criterion for different topics. By querying average rating

Figure 9: Business average rating on topics

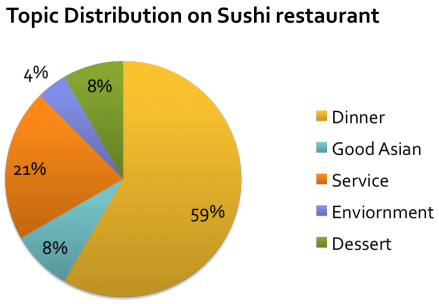

Figure 10: Topics distribution related to the sushi restaurant

on subtopics of a user, we can get an insight about this quickly. Here is an example for the same user we check in last problem (figure 8). This user has an average rating in property as 3.81. A obvious finding about this user is that he has a high standard for restaurant service as the average rating is as low as 2.6. Weighted factor based recommendation system may need to increase weighting for this topic.

5.3.4 Business hidden star rating

With dividing ratings of specific subtopics, restaurants could earn insights on how to improve their businesses. When predicting the star rating per hidden topic, we can get an insight for business owner that where need improvement and what is strength. Here, we query average star rating for with all reviews of one sushi restaurant who has an average rating as 3.5. We pick some typical subtopic to show in figure 9. The result shows that this restaurant has high rating on "environment" as , but bad review on "service". Compared to average rating of 3.5, we are curious there must be some parts of areas make this restaurant unsuccessful but users care much. In order to verify this assumption, we use another query to check distribution of user attention on each subtopics. From result in figure 10, we can realize that only few reviews talking about the topic that this restaurant achieves the best rating, "environment"; however, for "service" that the restaurant performs low score, nearly a quarter of users care about this topic. Moreover, it seems the restaurant need to perform better for dinner time since 58% reviews are about "dinner" but this restaurant only got an average rating of 3.6.

5.3.5 Usefulness prediction

Rank	Topic	Average votes
1	fast food	1.37
2	critism on service	1.33
3	special	1.32
4	food	1.26
5	ingredient	1.20
	...	...
30	Compliments	0.56

Table 6: Usefulness rank of topics: Top 5 and The last

Yelp website provides a function for user to vote for useful reviews. Yelp data challenge also proposes the problem "What makes a review useful" as a challenged analysis. By relate review usefulness votes to topic division, we can get a insight of average votes number of usefulness for each topic.

6. RECOMMENDATION MODELING

6.1 Latent-Factor Recommender Systems

Latent-factor recommender model [11] trains a model capable of predicting a score for each possible combination of users and items. The internal coefficients of the model are learned from known rating of users and items. Recommendations are then based on these scores. Formally, the predicted rating for user$i$ on item j is given by

$$f(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \gamma_i \tag{2}$$

where, α is an offset parameter, β_u and β_i are user and item biases, and γ_u and γ_i are K-dimensional user and item factors, where K is a hyperparameter to be tuned. Users and items are represented by weights and factors. Roughly speaking, the bias terms(β), account for a user or itemâĂŹs bias towards higher or lower ratings. For example, an item that is consistently rated highly would have a higher weight coefficient associated with them. Similarly, an item that consistently receives below average ratings would have a lower weight coefficient to account for this bias. The factor terms(γ) model interactions between users and items.

Given a training corpus of ratings , the parameters $\Theta = \alpha, \beta_u, \beta_i, \gamma_u, \gamma_i$ are learned by minimizing the squared error function:

$$\Theta = argmin_\Theta \sum_{u,i} (f(u,i) - r_{u,i})^2 + \lambda \Omega(\Theta) \tag{3}$$

where $\Omega(\Theta)$ is a regularizer that penalizes.
$\Omega(\Theta) = \sum_{u,i} ||\beta||_2^2 + \sum_u ||\gamma_u||_2^2 + ||\gamma_i||_2^2$

To deal with "cold start" problem, we can incorporate additional sources of information about the users and items to the base model above, so that the model can use implicit feedback to gain insight into user preferences or item characteristics.

$$f(u,i) = \alpha + \beta_u + \beta_i + a^T x_u + b^T y_i + \gamma_u + \gamma_i \tag{4}$$

where, x_i and y_j are respectively the user and item side feature vectors, and a and b are respectively the weight vectors for those side features. Accordingly, regularizer for linear factors need to be added into optimization function.

	Model	RMSE	Improve (to(2))	Improve (to(3))
(1)	Baseline(Offset+bias)	1.1221	-	-
(2)	Baseline(latent factors model)	1.1190	-	-
(3)	(2) + AvgRating + review_count	0.9692	13.37%	-
(4)	(3) + user votes	0.9690	13.41%	0.02%
(5)	User topics	0.9624	13.99%	0.70%
(6)	Business topics	0.9663	13.65%	0.30 %
(7)	User/Business topics	0.9607	14.15%	0.87%
(8)	User/Business topics + latent factor	0.9591	14.29%	1.04%

Table 7: RMSE of different recommendation model

6.2 Experiments and Result

We apply latent factor recommender to "Restaurant" data set, mentioned in Topic Modeling section. Then compare result of recommender with side feature and topic factors to baseline models: 1) Offset and bias only (α, β) 2) Basic latent factor recommender(eq. 2).

For topic factor model, we simply incorporate LDA topic prediction of topic probability vector to document with basic latent factor as users factor or business factor. LDA learns a set of topics(K=30) and topic proportions for each document. By treating each 'document' as the set of reviews for a particular product or user, LDA generates a stochastic vector, which we use to set γ. Further, we find that remaining additional latent factors can still improve the result in some degree.

Results in terms of the Root-Mean-Squared-Error(RMSE) are shown in Table 7. From the results, we can see that incorporating side features into recommendation model improves prediction performance significantly by 13.37%. Surprisingly, side feature of average rating of users and products reduce predict error in a large scale. In theory, basic latent factor model is supposed to capture rating feature automatically by bias and latent factors. We think there are two reasons causing the lower performance than expected of conventional latent factor model: 1) Cold start problem. Latent factor model can learn user feature, product feature and their interactive factors from former record, but in Yelp data set, as we mentioned in power law graph, user-review-business tuple is too scarce compared total amount of users and businesses. That means, there are large amount of users have few rating records for the model to learn so that new users can barely benefit from the model. 2) Parameter tuning. According to experiment, we find that hyper-parameters (value of regularizer and max iteration number) are crucial to final result. Another problem is for optimization, both baseline models converge slowly (SDG 300 iteration, 550 iteration) in constrain of best value of regularizer compared to new models with side feature. That makes train process complicated and lack of robust. In contrast, side features with fixed semantic meaning serve as a role of regression model. In summary, the general idea here is similar to Classifier Ensembles [],such as boosting classifier and learning forest, which are proved to outperform single classifier. Thus, we conclude that adding semantic side features is beneficial to recommendation prediction.

Even more, textual information represented by topic improve the model further by 1.04% (14.29% improvement to baseline). The result proves our assumption that review text contains large amount of information. Moreover, review text is a informative resource especially for cold start users or product. We can predict topics from one single review so that increase feature dimension for rating easily as we showed in last section. Subtopics from one review can still reveal a new user's preference or product genre, which release uncertainty of model caused by "cold start" problem. An interesting observation is that model with user's review topic performances better than model with business topic feature. A possible reason is that topics learned from Yelp reviews are more related to users' preference instead of business categories. This is reasonable since users tend to write review based on their criterion and expectations, but describe less about real categories of restaurants. In other words, topics clustered based on users' perspective may have less stable influence to rating on business side.

7. CONCLUSION AND FUTURE WORK

In this project, our work present that good feature engineering process is vital to machine learning model design. Particularly, side feature selected by linear regression in feature process and textual features extracted by topic mining model can improve recommender prediction significantly.

Besides, we believe topic modeling can also benefits from rating which can provide sentiment information for topic clustering. For example, in current LDA model, we may get a topic as service but it's hard to distinguish good service and bad service. That means rating information can be used in topic learning process and at the same time topic model can contribute to predictive model. Similar idea is proposed by McAuley's HFT and J.Linshi [13].

Further, as we described in topic modeling section, we can expect wide usage of the combining model to various predictive problems, like usefulness prediction.

8. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139, 1999.

[3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[6] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[7] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[8] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[9] J. Huang, S. Rogers, and E. Joo. Improving restaurants by extracting subtopics from yelp reviews. 2014.

[10] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 57–64. ACM, 2009.

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

[12] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[13] J. Linshi. Personalizing yelp star ratings: a semantic topic modeling approach.

[14] J. McAuley and J. 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.

[15] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908. International World Wide Web Conferences Steering Committee, 2013.

[16] O. Phelan, K. McCarthy, and B. Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388. ACM, 2009.