Utilizing Product Reviews to Predict Ratings for New Product

Bowen Zhu(bowenz), Mingzhi Zeng(mingzhiz)

1 Introduction

The goal of online recommendation system is generally to learn the potential connection between users and products, and provide the user with the most related products. For example, companies like Amazon make heavy use of the purchase histories from a customer group to promote add-on selling [3]. However, the existing approaches may fail to make the right recommendation when lacking information of a given product or user. This is the typical *cold-start* problem which happens frequently in commercial recommendation systems.

In this project we try to improve the quality of the recommendation when given only small amount of information about a certain product or a user. Specifically, we use the review text to build topic models for both products and users, and extract topic distributions as features for the Hidden Factor Model. We then measure the potential connection between products and users using the learned HFM model. The idea of using review texts comes from the fact that even a single review text may provide enough information for building and using the topic models.

This new approach to recommend products can be very useful in commercial recommendation systems. Especially when we need to do precise promotion for new products, or to attract newly registered customers. To measure the quality of the prediction methods, we take the MSE of the predicted score of each user-product pair as the metric. A lower MSE indicates a better performance.

We conducted experiments against two baseline algorithms. Our results showed a reduction of MSE comparing to the baseline algorithms in general settings and some cold-start settings.

2 Related Work

Much work has been done on recommender systems. To predict user ratings, Collaborative Filtering has always been used as the main approach. Collaborative Filtering is a method where items are recommended to a user based on other users with similar patterns of selected items. However, CF does not use any content information for items. There are two main methods used for CF, *latent factor models*[1] and *neighborhood methods*[5]. The first method considers every preference of user-item pair effected by hidden factors of user, item and the match between user and item. This model performed better than neighborhood method, so we will use latent factor models. But these models don't provide methods to use the content of items. Basically people use topic models to discover the topics from a large collection of documents. Here the topic is a distribution over terms under a certain theme. Though it is a low-dimensional representation for documents, it meet the need of most systems and has been used for many text-related recommendation tasks.

A simple topic model is *Latent Dirichlet Allocation (LDA)*, which was first developed by David Blei, Andrew Ng, and Michael Jordan in 2003[2]. A typical LDA model is a generative probabilistic model of a corpus

which generates K topic distributions of a fixed vocabulary. The basic process of LDA is as follows.

Step1. Draw topic proportions

$$\theta_i \sim Dirichlet(\alpha)$$
$$\beta_j \sim Dirichlet(\pi)$$

Step2. For each word j,
(a) Draw topic assignment $t_{ij} \sim Mult(\theta_i)$
(b) Draw word $w_{ij} \sim Mult(\beta_{t_{ij}})$
Where β is a certain topic.

There are also supervised LDA model (sLDA) and generalized LDA model (fLDA)[8][9]. Supervised LDA model emphasizes on the cases where documents are labelled with diverse responses. Another category of LDA is Discriminative LDA[6], which emphasizes on the problem that MLE might be suboptimal in the overall classification problem. In our task, we will focus on unsupervised LDA model.

By combining traditional collaborative filtering with topic modeling, collaborative topic regression (CTR) model simply fits the recommender system with text content[10]. The basic idea is to replace the latent item vector in latent-factor model with topic proportion. We will use this model and details are described later.

Julian McAuley and Jure Leskovec[7] also proposed an HFT model to improve the original CTR model. The new model adds corpus likelihood as the regularization term to get a better prediction result even in cold-start cases. Our algorithm also utilizes this idea to learn a prediction model that improves cold-start accuracy.

3 Data Set

The data set in use is the Amazon movie review dataset from Stanford SNAP group[4]. It contains 7,911,684 reviews, 889,176 users and 253,059 products. Each review in the dataset consists of product id, user id, profile name, helpfulness, review score, review time, summary and review text. We have downloaded it from Stanford website and preprocessed the information. To get tokens for building topic models, we also used NLTK tokenizer to tokenize the review text. Specifically, we processed 100,000 reviews from the dataset, then we built the corpus from the review text. Next we used LDA to extract Dirichlet distributions for topics and words and stored them as input features of the regression algorithms. These features will be characterized visually later in this report.

4 Recommendation Model

4.1 Data preprocessing

The basic idea to preprocess raw data is do statistics on the raw text and build a corpus that is well organized and representative for the whole data set. There are several potential approaches to do this. For example, we can use Principle Component Analysis to pick the top K words that are most representative of the entire vocabulary, and thus reduce the dimension of the problem. However, the matrix that is used as input for PCA is of dimension $|D| * |V|$ where $|D|$ is the size of the document set and $|V|$ is the size of the vocabulary. In our problem, the complexity of PCA is therefore $O(|V|^2)$, which is too high for a practical solution. Our later experiments also prove this conclusion.
We therefore applied several NLP techniques to build the corpus.

1) **Tokenization**: In this process we remove punctuations and stopwords. Specifically, we used the tokenizer from the NLTK package to tokenize the review text. Then we removed the stop-words such as “the”, “and” from the extracted tokens. Finally we filtered the tokens with punctuation inside. For Amazon dataset, we particularly add terms like “br”, “quot”, etc. to the stopwords list.

2) **Lemmatization**: We use WordNet lemmatizer to lemmatize the tokens in the review text. The goal of lemmatization is to resolve ambiguity and get uniform representation of the same word. This process increases the accuracy of frequency count.

TABLE 1
Selected removed tokens

Token	RSJ weight	Token	RSJ weight
movie	-0.4401	time	0.7990
film	0.4971	great	0.7735
one	0.2916	good	0.8320

Another preprocessing method on corpus is **Frequent Term Removal**. From our experiments, we notice that some terms appear in nearly all the product reviews. Adding these terms in topic distributions does no good in calculating topic distributions. Therefore we used a smoothed version of IDF score called RSJ weight, from BM25 retrieval algorithm, to judge if the term is a frequent term, and remove it from the corpus if needed. The RSJ weight is given by:

$$log\frac{nDocs - df + 0.5}{df + 0.5} \tag{1}$$

A list of selected removed tokens is shown in Table 1. Lemmatization and Frequent Term Removal turn out to have a significant impact on the topic words that the LDA algorithm picks. When these two techniques are not applied, stop words and punctuations will gain large weights, but they don't contribute to the topic distributions. A comparison between the MSE before and after applying these techniques is shown in Figure 1. From the figure we can see that by applying lemmatization and frequent term removal, we improve the MSE for all approaches.

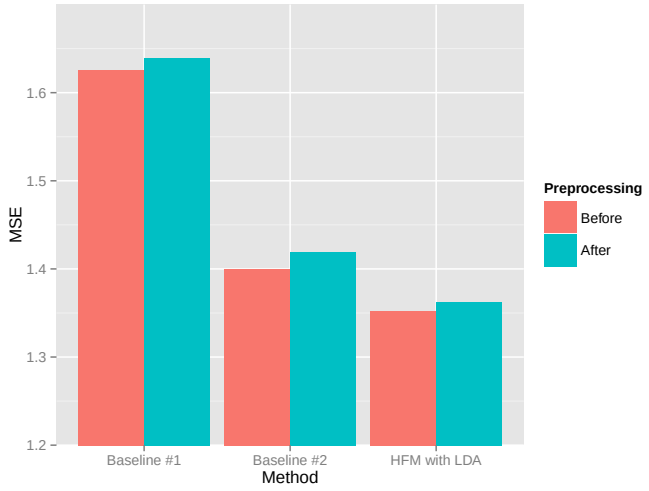

Fig. 1. *Comparison of MSE of different approaches before and after applying lemmatization and frequent term removal.*

4.2 Model analysis

Our system is a "Latent-Factor Recommender System" where the preferences for users and the features for movies are the latent-factors. A "standard" latent-factor model predicts ratings $r_{u,i}$ for user u and movie i as

$$r^*_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \tag{2}$$

Where the α is an offset parameter for all users and all movies, β_u and β_i are individual offsets for different users and different movies, these parameters show how they biased to the average properties. γ_u and γ_i are both K-dimensional vectors representing the preferences for each user and the features for each movie. Here K is the number of topics. So a big $\gamma_u \cdot \gamma_i$ means the user's preference highly matches the movie's feature then we may predict a high rate on this match.
Next we will do regression on this model and get the parameters. In other word, our goal is to minimize the Mean Squared Error

$$\frac{1}{N}\sum_{r_{u,i}\in R}(r^*_{u,i} - r_{u,i})^2 \tag{3}$$

Where R is the set of ratings with the total number of N.
To transform the reviews into γ_u and γ_i, we use LDA model to find the latent topics uncovered in the reviews. This can tell us the probability of each topic for a movie $\theta_{i,k}$ according to the words distribution. So we can get γ_i with LDA and then do the collaborative topic regression. To solve the cold-start problem, we add a regularization term for the regression model. Instead of L1 or L2 regularization term, we want to add more information about the distribution for topics among documents and words so we come up with the *corpus likelihood term*. The

term is shown in equation below.

$$p(\tau \mid \theta, \phi, z) = \prod_{d \in \tau} \prod_{j=1}^{N_d} \theta_{d,z_{d,j}} \phi_{z_{d,j},w_{d,j}} \quad (4)$$

The presence of this term is critical even if we just want to predict ratings. The corpus likelihood acts as the regularizer for the rating prediction model and will pushing γ_i and γ_u toward zero and reduce bias when there are few reviews and ratings. So the final regression model is

$$\frac{1}{N} \sum_{r_{u,i} \in R} (r^*_{u,i} - r_{u,i})^2 + \lambda p(\tau \mid \theta, \phi, z) \quad (5)$$

Here λ is a hyperparameter that trades-off the importance of these two effects.
So the final graphical model is shown as Figure 2. The left part is a traditional LDA model. I means the hidden factors for movies and u stands for those factors for users. So in our model movie factors are influenced by topic-document distribution.

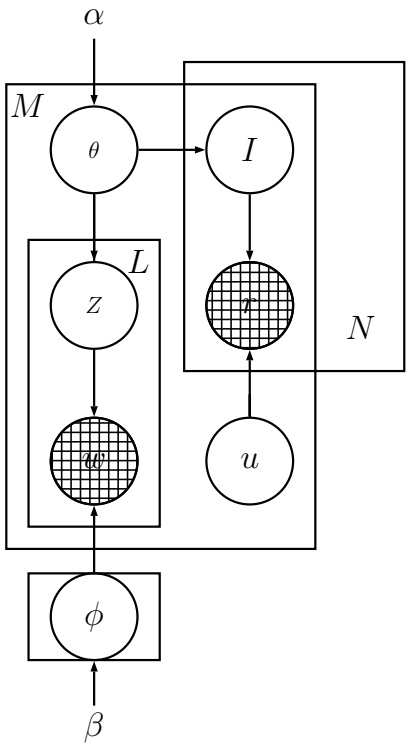

Fig. 2. *Graphical Model for HMF.*

4.3 Parameter learning

The process of learning parameters includes two main steps. First updates the parameters by minimizing the regression error function. Then use the distribution we have to resample the topic assignments for words and update z. Details described below.

Step 1 : Update $\alpha^t, \beta^t, \gamma^t, \theta^t, \phi^t$ *by* z^{t-1}

Step 2 : Sample $z^t_{d,j}$ *with* $p(z^t_{d,j} = k) = \phi_{k,w^t_{d,j}}$

For the optimization part, there are several methods to solve this. We choose gradient decent to estimate those parameters. The update rules for each parameter are as follows.

$$\alpha' = \frac{1}{N} \sum_{r_{u,i} \in R} 2(r^*_{u,i} - r_{u,i}) \quad (6)$$

$$\beta'_u = \frac{1}{N} \sum_{r_{u,i} \in R_u} 2(r^*_{u,i} - r_{u,i}) \quad (7)$$

$$\beta'_i = \frac{1}{N} \sum_{r_{u,i} \in R_i} 2(r^*_{u,i} - r_{u,i}) \quad (8)$$

$$\gamma'_u = \frac{1}{N} \sum_{r_{u,i} \in R_u} 2\gamma_i(r^*_{u,i} - r_{u,i}) \quad (9)$$

(10)

$$\gamma'_i = \frac{1}{N} \sum_{r_{u,i} \in R_i} 2\gamma_u(r^*_{u,i} - r_{u,i}) + \sum_{j=1}^{N_i} log(\theta_{i,z_{i,j}} \phi_{z_{i,j},w_{i,j}}) \quad (11)$$

Where R_u is the rating set for user u and R_i is the rating set for movie i. In our algorithm, the initial value of γ_i comes from LDA model. As we described in the previous part, LDA brings out the K-dimensional topic distribution θ_i for each movie. Instead of assigning this vector to γ_i directly, we should do normalization for each θ_i first to fit the latent-factor model.

5 Evaluation

5.1 Experiment

We used 100,000 reviews as the whole dataset, 80% of them as training data and the rest as testing data. We built two baseline algorithms

to compare with our algorithm.

Baseline#1: Estimate only α in equation (2). That means we use the average rating of the training ratings to predict all ratings for every user-movie pair.

Baseline#2: Estimate all parameters including γ_i in equation (2) with gradient decent. The initial values for those parameters are set as follows.

$$\alpha_0 = \frac{1}{N} \sum_{r_{u,i} \in R} r_{u,i} \tag{12}$$

$$\beta_{u0} = \frac{1}{N_u} \sum_{r_{u,i} \in R_u} r_{u,i} - \alpha \tag{13}$$

$$\beta_{i0} = \frac{1}{N_i} \sum_{r_{u,i} \in R_i} r_{u,i} - \alpha \tag{14}$$

Besides, γ_u and γ_i are set as random vector with each value between 0 and 1. The learning rate is 0.2 and stop criteria is $\mid f(x)' - f(x) \mid < 0.00001$.

HFM+LDA: We use a well-built LDA toolbox to extract the topic from the preprocessed data. The number of topics is set to $K = 10$. Then we use the normalized topic distribution for each movie i as the γ_i. The rest is as the same as baseline #2. So the difference between this algorithm and the baseline #2 is that we treat the movie features as constant when doing gradient decent. We make use of the reviews to help us make prediction but not just relying on ratings.

HFM+LDA with regularization: Add the regularization term and learn parameters as described in the previous part. In this model, we do learn the hidden factor γ_i with the initial value set by LDA. The tradeoff parameter $\lambda = 0.5$.

We evaluate these four algorithms by MSE in two situations (general and cold-start) and get the result as showed in table 2 and table 3. As we can see, the trivial baseline #1 has the biggest MSE because this method treats all users and movies as the same. Baseline #2 performs better than baseline #1 since this method treats users and movies differently with the latent-factor model and optimize them together. HFM+LDA model, which shows a smaller MSE than two baseline methods, effectively uses the knowledge from the review text and predicts ratings through topic matches. HFM+LDA with regularization model, which shows the smallest MSE among the all four methods. In general cases we can't see much improvement by adding this regularization term. But in cold-start case, this algorithm shows better performance than HFM+LDA model. The MSE for cold-start problem is relatively high. But it's not a surprise because just few ratings and reviews can be used in cold-start problem. We also tried different topic numbers K to test how this value affects the performance. The result is shown in Figure 3. We find that the MSE is lowest when topic number $K = 8$ and then raises when K increases. But our algorithm always gets a lower MSE than the baseline #2 in all test K.

TABLE 2
MSE of different algorithms in general tests

Baseline #1	Baseline #2	LDA	LDA with Regularzaition
1.6273	1.5549	1.4978	1.4974

TABLE 3
MSE of different algorithms in cold-start tests

Baseline #1	Baseline #2	LDA	LDA with Regularzaition
2.8761	2.8313	2.7633	2.7483

5.2 Visualization

1) **Topic Distribution**
To get a direct interpretation of what the features we have extracted are, we model the Dirichlet distributions with a two-dimensional heatmap, as shown in Figure 4 and Figure 5. Column labels are topics and the row labels are the words or documents occurred in the topics. Each unit in the heatmap indicates a topic-word pair or topic-document pair. Dark green indicates higher frequency

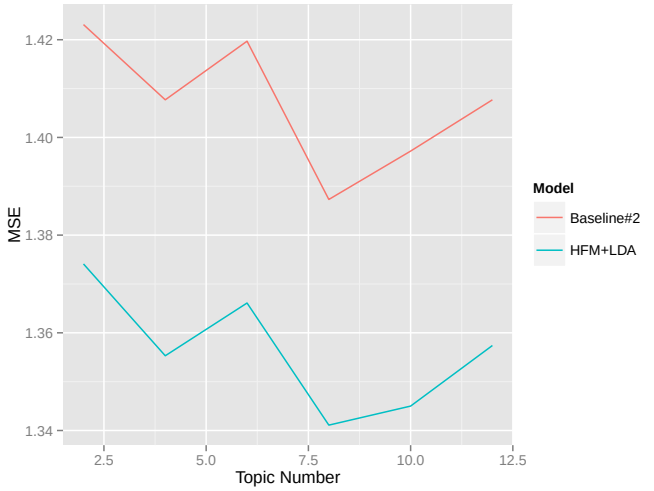

Fig. 3. *MSE for different topic number K.*

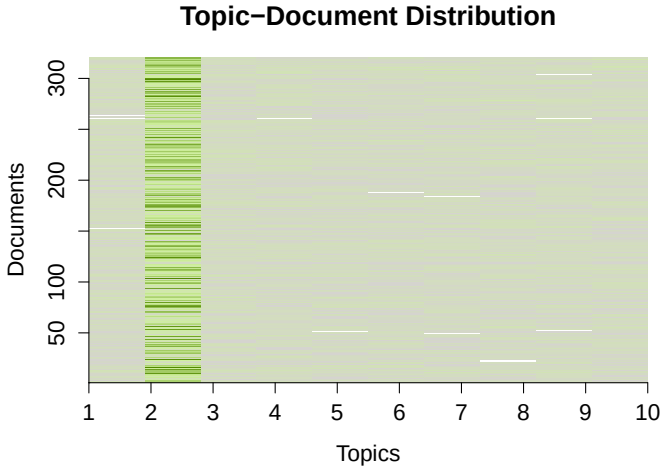

Fig. 5. *Topic-Document distribution in 10,000 Amazon reviews.*

and light green indicates lower frequency.

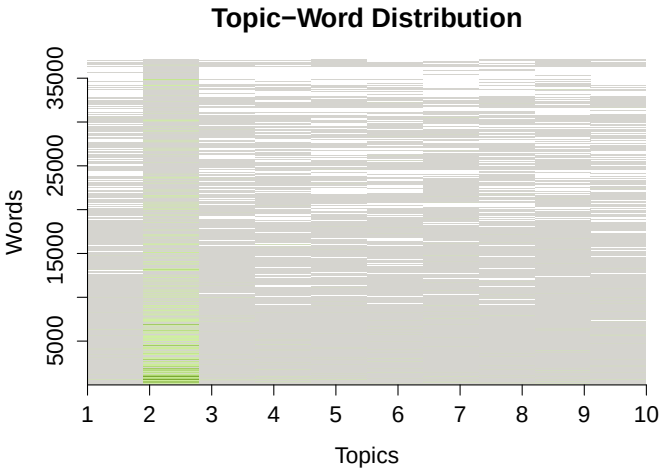

Fig. 4. *Topic-Word distribution in 10,000 Amazon reviews.*

From the figure we can conclude that some terms may occur in many topics, and with a large proportion. This kind of distribution will potentially harm the performance of the topic regression because, in this way, many topics will share the same terms and thus will have less distinguishing power.

2) **Topic Co-occurrence**
Another view of the topics is the co-occurrence connection graph. Shown in Figure 6. A node represents a topic, the label of the node is the terms occurred in the topic, and the edge and edge weight stand for the co-occurrence of topics and the corresponding frequency.
What we can conclude from this figure is a little bit different from Figure 4. In this figure we can easily find that one topic tends to be very centralized in the topic connection. The intuitive interpretation for this is this topic might have appeared in almost all reviews. This will also harm the performance of the topic regression algorithm because, if the topic distribution of all documents became centralized towards certain topic, then according to the Latent Factor Model, other topic features in γ_i will be overwhelmed by the big weight of this topic. Threfore the features we extracted will be weaker in distinguishing products.
The improvement will be discussed in conclusions.

6 Conclusions

We built an LDA model to extract topic distributions from raw review text, which helped modeling the characteristics that other approaches often ignore. We have also implemented the Hidden Factor Model to utilize these features to make score predictions. From the results, our LDA-HFM model

Fig. 6. *Topic connections in 10,000 Amazon reviews.*

outperforms the two baselines in general situations. And by adding regularization terms, our model can beat the two baseline algorithms as well as the model without regularization terms, in some cold-start settings.
There are still some work to do in the future.

Document Representation: Now our reviews are represented by BOW model and the terms are not so "clean". To get a high accuracy on topic modeling, a good feature extraction is needed. We can try using n-gram model to add phrases into the vocabulary. Also, in this problem, the sentiment words like "happy", "annoyed" will be more important than other words. Traditional feature extraction method may filter these words which is not what we expect. So we can use some sentiment algorithms or some other advanced NLP algorithm to get a better feature extraction.

Algorithm Improvement: Now we are using unsupervised LDA model to learn the topics inside the documents. We can try some supervised model like sLDA to use ratings as the labels to train a new topic model and then apply it in our algorithm. We can also generate topic seeds first then use them as the labels for LDA. Finally, in our algorithm, we just group the reviews according to movies but not use them to estimate γ_u. So we can also make use of the reviews by same user. Some approaches like fLDA can estimate the hidden factors of both users and movies at the same time. In the previous part we have discussed about the centralization behavior of both topic distributions and word distributions. The improvement of model may solve this problem.

System Performance: Our current implementation used only 100,000 reviews from the dataset due to the time and space limitation. The limitation lies mostly in the LDA part. Since we need to do experiment on a larger dataset in the future, we can continue

optimizing the performance of our code.

7 Acknowledgment

We would like to send our thanks to Abul Saparov, who gave us many insightful suggestions on the topic model. We would like to thank all the students and instructors who shared their knowledge with us during the poster session.

References

[1] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

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

[3] Anand V Bodapati. Recommendation systems with purchase data. *Journal of Marketing Research*, 45(1):77–93, 2008.

[4] Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of the 22nd international conference on World Wide Web*, pages 307–318. International World Wide Web Conferences Steering Committee, 2013.

[5] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[6] Simon Lacoste-Julien, Fei Sha, and Michael I Jordan. Disclda: Discriminative learning for dimensionality reduction and classification. In *Advances in neural information processing systems*, pages 897–904, 2009.

[7] 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.

[8] Jon D Mcauliffe and David M Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.

[9] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[10] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.