A Tensor Factorization on Rating Prediction for Recommendation by Feature Extraction from Reviews

Yang Sun
Graduate School of Engineering,
Hiroshima University, Japan
sunyang@se.hiroshima-u.ac.jp

Guan-Shen Fang
National Institute of Technology,
Tsuyama College, Japan
bou@tsuyama-ct.ac.jp

Sayaka Kamei
Graduate School of Engineering,
Hiroshima University, Japan
s-kamei@se.hiroshima-u.ac.jp

***Abstract*—In many online review sites or social media, each user is encouraged to assign a numeric rating and write a textual review as a feedback to each item that he had gotten, e.g., a product that he had bought, a place that he had visited, a service that he had received. Sometimes, feedbacks by some users would be affected by some contextual factors such as weather, distance, time and season. In addition, the context-aware approach is being developed by utilizing the user's contextual information to produce more precise recommendations than traditional approaches. Furthermore, previous works [1], [2] have already approved the drawback of the ignorance of textual reviews would bring mediocre performance for rating prediction. In this work, we propose two rating prediction models TF-Doc2Vec and TF-LDA. Tensor Factorization (TF) is an extended version of Matrix Factorization (MF) by adding another dimension. We consider seasonal context as the additional dimension. Firstly, in TF-Doc2Vec (resp. TF-LDA), each of the reviews is characterized by a numeric feature vector via Doc2Vec (resp. a topic distribution via LDA). Secondly, TF-Doc2Vec uses TF which is trained by the proposed first-order gradient descent method for TF, named Feature Vector Gradient Descent (FVGD), while TF-LDA uses TF which is trained by extended TGD proposed in [1]. In our evaluation, we use pre-processed data of three cities in the YELP challenge dataset. We conduct experimental comparisons, and results show that TF-Doc2Vec and TF-LDA improve the performance significantly as compared to the basic TF model.**

***Index Terms*—Rating prediction, Tensor Factorization, Doc2Vec, LDA, Recommendation System**

I. Introduction

Nowadays, recommender systems are an essential part of online services and social networks. The system would give recommendations to users by using feedback from users. The feedbacks are offered by the users after their purchases or experiences, and each of them includes a numeric rating and a textual review as the evaluation. Also, the reviews have some relationships with ratings to some extent. From reviews, we can roughly understand why users give such ratings for some items.

Collaborative Filtering (CF) method is considered as the most effective and widely-used method to predict ratings. According to the similarity of preferences of the neighborhoods, we can calculate a numeric value that can represent whether a user would like the item or not. In spite of the high accuracy that the CF method can offer, the data sparsity still causes a big effect for first-used users that contain very fewer information.

Among all the CF methods, the Matrix Factorization (MF) is an ideal method to predict ratings [3]. However, recent researches pointed out that the ignorance of the context is the major shortcoming of basic MF. Setiowati *et al*. [4] published a survey paper to point out that the context-aware approach is being developed by utilizing contextual information to produce more precise recommendation according to user's preferences. The results of some studies [2], [5]–[7] have shown that the implementation of context-awareness on a recommendation system has given better results for personalized recommendations rather than the systems without it. Additionally, recent researches [1], [2] pointed out that the ignorance of the reviews is the major shortcoming of basic MF and brings it mediocre performance.

In this paper, in order to offer a high quality of recommendation, we propose new methods TF-Doc2Vec and TF-LDA to predict the rating on user's purposeful POI for the recommendation with considering seasonal context factors by extending Tensor Factorization (TF) [8]. Firstly, for TF-Doc2Vec, we train Doc2Vec model to obtain distributed representations of reviews. We assume that each distributed representation of a review by a user for a POI represents weights distribution of each latent factor to represent how much the user shows interest to the POI. That is, in this method, against each visited POI, each of the users would be assigned a unique feature vector with numeric values via Doc2Vec. After that, this method learns the feature vectors of users by the proposed first-order gradient descent method, called Feature Vector Gradient Descent (FVGD). For TF-LDA, we obtain the topic distributions of reviews by Latent Dirichlet Allocation (LDA). By extending Topic Gradient Descent (TGD) [1], this method learns the topic distributions of users.

In our evaluation, we conduct experiments via using data about restaurants in three cities in YELP challenge dataset. We compare the performance of the following six methods: basic MF method, MF-LDA by [1], MF method with FVGD (MF-Doc2Vec), basic TF method, TF-LDA, and TF-Doc2Vec. Meanwhile, we use Mean Absolute Error (MAE) and Root

Mean Square Error (RMSE) performance indicators to evaluate which method would get the best result.

The contribution of this paper is as follows:

- We use a more complex TF model to predict ratings than traditional (basic) MF method and TF method. Not only users and items (POIs) are corresponding to the latent factor but also the season would have some relationships with the latent factor to some extent. To consider textual review contents, we propose new two gradient descent methods. One is FVGD for using distributed representations obtained by Doc2Vec, and the other is the extended TGD for using topic distributions obtained by LDA.
- In the evaluation, we compare the performance of our methods with other four approaches in rating prediction. The result obviously shows that the performance of three context-aware TF-based methods is better than three MF-based methods respectively. In addition, proposed methods expressed better performance than the basic TF method.

The remainder of this paper is organized as follows: Section II overviews related works of MF and TF models. Section III gives the problem definition and the description of the basic MF and TF models. Section IV describes the existing method, i.e., MF-LDA. Section V describes TF-LDA and TF-Doc2Vec. Section VI represents the experiment to compare six methods including proposed methods. Finally, section VII concludes the paper with future work.

II. Related Work

In recent years, researchers have paid more attention to the MF and TF methods because it can solve the data sparsity problem of CF methods [9]. Salakhutdinov *et al*. [10] proposed the probabilistic MF and introduced Gaussian priors as hyper-parameters to present latent factors. They noted that maximizing the log-posterior of the ratings over users' and items' latent factors is equivalent to minimizing the squared errors of the rating prediction.

Against the ignorance of the contextual information in the MF and TF methods, there are various approaches. Karatzoglou *et al*. [11] proposed a model, called multiverse recommendation, in which different types of context are considered as additional dimensions in the representation of the data. This method can address the *N*-dimensional factorization. And the result has shown that their method improved upon non-contextual basic MF up to 30% in terms of the MAE. Yao *et al*. [12] proposed the non-negative TF method that exploits a high-order tensor instead of the traditional user-location matrix to model multi-dimensional contextual information. Experimental results on real-world datasets demonstrate it has higher accuracy than basic MF.

To consider textual reviews in the MF model, [13] and [14] proposed methods with the transformation of topic distribution of reviews to latent factors of MF. Although their methods outperform the basic MF, the drawback of their methods is their complexity. In response, Fang *et al*. [1] proposed a novel method MF-LDA of rating prediction which uses both the ratings and reviews, including a new first-order gradient descent method for MF, named Topic Gradient Descent (TGD). This method binds the latent topics by LDA to latent factors via the training process of MF (Thus, TGD is without the complicated transformation. We'll explain more details about this in section IV.). Comparing with methods in [13] and [14], the performance of MF-LDA increased.

III. Preliminaries

A. Problem Definition

Each feedback includes a numerical rating in scale of $[1, 5]$ and a textual correlated review. We consider the season as a context for POI recommendation. Thus, we suppose, in a set of feedbacks, we have I users, J items (POI) and S seasons. So, in a feedback, the rating made by user $u_i (i \in \{1, \ldots, I\})$ to item $v_j (j \in \{1, \ldots, J\})$ under season $c_s (s \in \{1, \ldots, S\})$ is denoted as $r_{i,j,s}$. If $r_{i,j,s}$ exists, it must have a correlated review $d_{i,j,s}$ written by u_i. Therefore, the feedback is a 5-tuple $< u_i, v_j, c_s, r_{i,j,s}, d_{i,j,s} >$. Then, we consider the problem to predict a missing rating $\hat{r}_{i,j,s}$ [1] for a given user u_i and a given item v_j under season c_s.

B. Matrix Factorization for Recommendation

MF is regarded as an ideal method to predict ratings. In this subsection, we describe the basic MF and biased MF, which are used in this paper.

Biased MF [3] is also an influential method to predict the missing ratings based on basic MF. At first, it will initialize two predefined arbitrary matrices with *K* dimensional latent factor space. Accordingly, one is user matrix U in which each vector $U_i \in \mathbb{R}^K$ is associated with a user u_i. The elements of matrix measure the user's extent of interest to such factors. The other matrix is item matrix V in which each vector $V_j \in \mathbb{R}^K$ is associated with an item v_j. The vector V_j presents the positive or negative extent of those factors that v_j possesses. The inner product of U_i and V_j represents the interaction of u_i and v_j, and approximates the corresponding rating $r_{i,j}$ as follows:

$$r_{i,j} \sim \hat{r}_{i,j} = U_i^T V_j + \mu + b_i + b_j, \tag{1}$$

where μ is the average of ratings overall users and items, and b_i and b_j denote the observed biases of user u_i and item v_j, respectively. The object of equation (1) is to learn U_i and V_j by given training set including real ratings, by minimizing the loss function as follows:

$$\pounds = \frac{1}{2} \sum_{i,j} [(r_{i,j} - \hat{r}_{i,j})^2 + \lambda(||U_i||^2 + ||V_j||^2 + b_i^2 + b_j^2)], \tag{2}$$

where λ is the parameter to control the regularization to avoid over-fitting in learning. A typical way to minimize the objective function (2) is to use a gradient descent algorithm. It calculates the gradients of U_i and V_j for every given rating $r_{i,j}$ as follows:

$$\begin{aligned} gU_i &= -(r_{i,j} - \hat{r}_{i,j})V_j + \lambda \cdot U_i, \\ gV_j &= -(r_{i,j} - \hat{r}_{i,j})U_i + \lambda \cdot V_j. \end{aligned} \tag{3}$$

[1] There is no rating $r_{i,j,s}$ by u_i about v_j under c_s in the dataset.

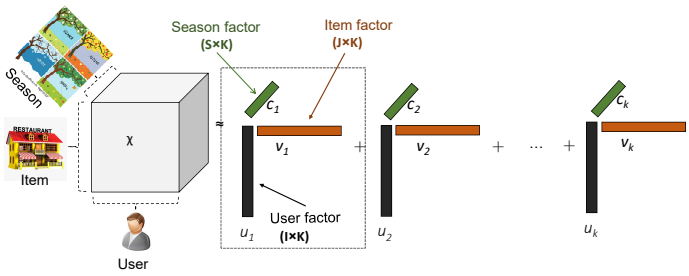

Fig. 1. CP tensor decomposition

We can get the basic MF by deleting the terms of μ, b_i and b_j from the equations (1) and (2).

C. Tensor Factorization for Recommendation

The CP-tensor decomposition (CP-TD) is one of TF algorithms [15], [12]. The objective of TF is to predict user ratings for some items under a contextual condition. Comparing to the MF, the TF can be considered as a generalization of MF that allows for flexible and generic integration of contextual information by modeling the data as a User-Item-Context K-dimensional tensor instead of the traditional 2D User-Item matrix.

In Fig.1, we show a three-dimensional space, i.e. a tensor χ, as a predicted rating model for recommendation system. It maps users, items, and contexts into a joint latent factor space with K dimensions where K is arbitrarily predefined. Comparing to the MF in section III-B, we set the same size of K dimensions for latent factor in TF. Differently, C is another contextual matrix whose vector $C_s \in \mathbb{R}^K$ is associated with a given context c_s. Each element $r_{i,j,s}$ of the tensor χ on the position index (i, j, s) is approximated as follows:

$$r_{i,j,s} \sim \hat{r}_{i,j,s} = \sum_{k=1}^{K} u_{ik} \cdot v_{jk} \cdot c_{sk}. \tag{4}$$

The objective of CP-TD is also to minimize the function of regularized squared error:

$$\mathcal{L} = \frac{1}{2} \sum_{(i,j,s)} (r_{i,j,s} - \hat{r}_{i,j,s})^2 + \frac{\lambda}{2} \sum_{k=1}^{K} (||U_i||^2 + ||V_j||^2 + ||C_s||^2), \tag{5}$$

where λ is the parameter to control the regularization to avoid over-fitting in learning, and $||\cdot||^2$ denotes the L^2 norm.

A similar way to minimize the objective function (5) is also to use stochastic gradient descent. It calculates the gradients of U_i, V_j and C_s for every given rating $r_{i,j,s}$ as

$$\begin{aligned} gU_i &= -(r_{i,j,s} - \hat{r}_{i,j,s}) V_j \circ C_s + \lambda \cdot U_i, \\ gV_j &= -(r_{i,j,s} - \hat{r}_{i,j,s}) U_i \circ C_s + \lambda \cdot V_j, \\ gC_s &= -(r_{i,j,s} - \hat{r}_{i,j,s}) V_i \circ U_j + \lambda \cdot C_s, \end{aligned} \tag{6}$$

where $\circ$ represents Hadamard product between two vectors, and updates three gradients to the inverse direction of gradient iteratively. The updating step is often unique and controlled by a constant learning rate. So the objective would be converged with a proper learning rate.

IV. Existing Method

In this section, we explain an existing method MF-LDA proposed by Fang *et al.* [1]. It is a novel method of rating prediction which uses both the ratings and reviews, and includes a new first-order gradient descent method for MF, named TGD. It is based on the biased MF.

In MF-LDA, they derive topic distribution from user's reviews via LDA. LDA is a probabilistic generative latent topic model of a set of semantic documents called corpus. Its idea is that each latent topic is characterized by a distribution over words, and a document is a random mixture over such topics. MF-LDA binds the latent topics to latent factors via the training process of MF instead of the complicated transformation as in [13] and [14].

With a given set of the history of feedbacks, the first task is to derive the topic distribution from each review by LDA. Each review in the feedbacks was regarded as a single document and all reviews as the corpus D. Assume that there are K topics overall in D, which are shared by all documents. A topic is denoted by t_k with $k \in \{1, \ldots, K\}$. For a review $d_{i,j} \in D$ to item v_j by user u_i, its topic distribution is denoted by $\theta_{i,j}$, which is a K-dimensional stochastic vector. Therefore, each of the elements $\theta^k_{i,j}$ represents the proportion of corresponding topic t_k having been mentioned in $d_{i,j}$.

The next step is to model the ratings using MF and further to predict the ratings for users. The difficulty comes from the link of the topic distributions of reviews and latent factors without a complicated transformation between them. So the TGD method is to correlate them through the training process of MF. Since the reviews provide an efficient tool for the users to explain their ratings, important topics are often mentioned much in the reviews. The key idea is to use $\theta^k_{i,j}$ to affect the learning of U_i and V_j in the training process of MF. With the denotation of gradients gU_i and gV_j in equation (3), the updating equations for U_i and V_j are

$$\begin{aligned} U_i &\leftarrow U_i - \gamma_M H^\theta_{i,j} \cdot gU_i, \\ V_j &\leftarrow V_j - \gamma_M H^\theta_{i,j} \cdot gV_j, \end{aligned}$$

where γ_M is a pre-defined constant for MF model, and $H^\theta_{i,j}$ is a $K \times K$ diagonal matrix such that the k-th diagonal element is $\theta^k_{i,j}$. $H^\theta_{i,j}$ is together with γ_M to be the learning rate, which assigns various updating steps for each latent factor.

With the MF model trained by TGD, for a given user u_i and an unpurchased item v_j, we calculate the rating prediction $\hat{r}_{i,j}$ following equation (1).

V. Proposed Method

In this section, we propose two methods to predict missing ratings. Both of our methods have a similar structure shown in Fig. 2. In TF-Doc2Vec, Doc2Vec is used for feature extraction to get feature vectors of reviews. In TF-LDA, for each review, its topic distribution via LDA is used as the feature vector. After that, based on the feature vector, both methods train TF models using new gradient descent methods respectively. For TF-Doc2Vec, we propose a new first-order gradient descent

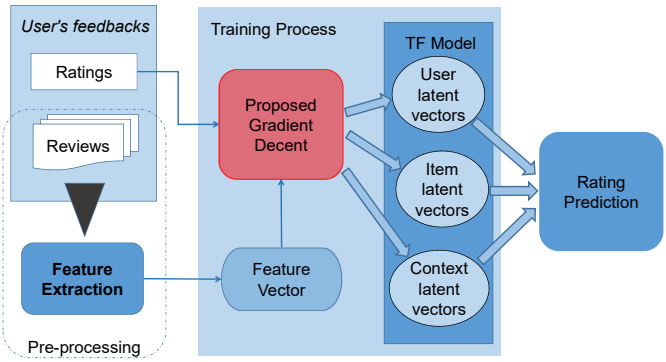

Fig. 2. The construction of proposed methods

method named Feature Vector Gradient Descent (FVGD), and for TF-LDA, we extend TGD [1].

A. TF-Doc2Vec

In this subsection, we illustrate the detail of our proposed method TF-Doc2Vec. From a set of user's reviews, the first task for TF-Doc2Vec is to pre-process the corpus D. We can regard each review by user u_i to item v_j in season c_s as a document $d_{i,j,s} \in D$. Then we can use Doc2Vec, which is a technique to convert a document $d_{i,j,s}$ to a numeric vector $\delta_{i,j,s}$ called a feature vector. In each feature vector $\delta_{i,j,s}$, each element is weight $\delta^k_{i,j,s}$ corresponding to the latent factor $k \in \{1, \ldots, K\}$. The size of the corresponding latent factor K is determined by the size of the feature vector. Then, $\delta^k_{i,j,s}$ represents how much interest a user u_i shows to a latent factor k of item v_j in season c_s. Following the method proposed in [16], we train Doc2Vec model for D and infer feature vector from each review $d_{i,j,s}$.

Next, we use a TF model to predict the missing rating values for users. TF-Doc2Vec integrates the feature vectors of reviews and latent factors. So we propose FVGD to correlate them during the training of TF. We assume that the feature vector $\delta_{i,j,s}$ represents the importance of the degree of features in the evaluation of user u_i to item v_j under context c_s. Assume that the number of latent factors of TF is equal to K. The key idea is to make $\delta^k_{i,j,s}$ affect the learning of TF. With the denotation of gradients gU_i, gV_j and gC_s in equation (6), we write the updating equation for U_i, V_j and C_s as

$$U_i \leftarrow U_i - \gamma_T H^{\delta}_{i,j,s} \cdot gU_i,$$
$$V_j \leftarrow V_j - \gamma_T H^{\delta}_{i,j,s} \cdot gV_j,$$
$$C_s \leftarrow C_s - \gamma_T H^{\delta}_{i,j,s} \cdot gC_s,$$

where γ_T is a pre-defined constant which is the original learning rate that we initialize a proper value for TF firstly before the training process started. $H^{\delta}_{i,j,s}$ is a $K \times K$ diagonal matrix such that the k-th diagonal element is $\delta^k_{i,j,s}$. $H^{\delta}_{i,j,s}$ is together with γ_T to be the new learning rate which assigns various updating steps for each latent factor.

For the features which have high importance and generate much error, their corresponding latent factors are updated with large steps in every epoch of training. In contrast, factors of unimportant features are updated with small steps in every epoch of training. When U_i, V_j and C_s are initialized with vectors of extremely small constant, such factors will remain the initial values and further have little impact on the rating prediction. Then, by using the TF model trained by FVGD, we can calculate the rating prediction $\hat{r}_{i,j,s}$ following equation (4).

Note that, we can apply FVGD to the biased MF model by using Doc2Vec. That is, we can get the feature vectors $\delta_{i,j}$ via Doc2Vec, and use the following equations:

$$U_i \leftarrow U_i - \gamma_M H^{\delta}_{i,j} \cdot gU_i,$$
$$V_j \leftarrow V_j - \gamma_M H^{\delta}_{i,j} \cdot gV_j,$$

where $H^{\delta}_{i,j}$ is a $K \times K$ diagonal matrix such that the k-th diagonal element is $\delta^k_{i,j}$. We call it MF-Doc2Vec. With the MF model trained by FVGD, for a given user u_i and an unpurchased item v_j, we calculate the rating prediction $\hat{r}_{i,j}$ following equation (1).

B. TF-LDA

In this subsection, we illustrate another proposed methods TF-LDA. By extending TGD, we can simply extend MF-LDA to a TF method. That is, we also use LDA to derive the topics from user's review. Then, we can get the topic distribution $\theta_{i,j,s}$. With the denotation of gradients gU_i, gV_j and gC_s in equation (6), the updating equations for U_i, V_j and C_s are

$$U_i \leftarrow U_i - \gamma_T H^{\theta}_{i,j,s} \cdot gU_i,$$
$$V_j \leftarrow V_j - \gamma_T H^{\theta}_{i,j,s} \cdot gV_j,$$
$$C_s \leftarrow C_s - \gamma_T H^{\theta}_{i,j,s} \cdot gC_s,$$

where γ_T is a pre-defined constant for TF model, and $H^{\theta}_{i,j,s}$ is a $K \times K$ diagonal matrix such that the k-th diagonal element is $\theta^k_{i,j,s}$.

VI. Evaluation

A. Datasets and Implementation

We selected three cities data from *YELP challenge dataset*. YELP dataset contains many types of business, such as supermarket, salon, event, and restaurant. Here, we only use data with "restaurant" label for our experiment because we consider the feedback for restaurants reflects the change of seasons. After that, we filter out the users and items with less information in order to make sure the quality of results. The filtering constraints are as follows:

- Each review contains at least 10 words,
- Each of the users has at least 5 feedbacks under at least one season, and
- Each of the items concerns with at least 5 feedbacks under at least one season.

Following, we select the cities whose data sparsity and whose average number of words in reviews are as large as possible. Then, we only independently utilize the feedbacks from the state of Richmond Hill, Champaign, and Peoria from the dataset. For each review, we discard the stop words,

TABLE I
THE DESCRIPTION OF CITIES OF YELP USED IN EXPERIMENTS

city	#feedbacks	avg.rating	#users	#items	sparsity	avg.words
Richmond Hill	2336	3.4486	171	122	0.112	79.6
Champaign	2824	3.7178	203	104	0.134	61.9
Peoria	3116	3.6874	324	152	0.063	57.5

repeated words and punctuations, and do the stemming. With these pre-processes, TABLE I shows their statistics including the number of feedbacks, the average of ratings, the number of users, the number of items, the sparsity and the average of the number of words in reviews. The sparsity of a dataset is calculated as #feedbacks/(#users $\times$ #items). For each dataset, we use 5-fold cross validation that randomly takes 80% of the feedbacks as the training set and the rest as the test set to conduct the experiments.

According to month, we set four seasons $C_s = \{\text{Spring, Summer, Autumn, Winter}\}$ for TF models as follows:

- Spring: March, April, May
- Summer: June, July, August
- Autumn: September, October, November
- Winter: December, January, February

As a comparison, we train basic MF, MF-LDA, MF-Doc2Vec, and basic TF with the same parameter values as TF-Doc2Vec and TF-LDA in order to guarantee fairness. For each of the training sets, we train each model to observe the sum of squared error of rating prediction in each epoch. We test by changing parameter K from 5 to 60, and fixed regularization λ to 0.0005. As for the learning rate γ_M and γ_T, it is much easier to converge with a smaller learning rate. So we set $\gamma_M = \gamma_T = 0.0002$ for our following experiments. The latent factors in U_i, V_j, and C_s are initialized by randomly generated values following uniform distribution over $[0, 1)$.

In order to determine which method would achieve the best result, we need to select some performance indicators to evaluate. For the problem to predict the ratings, the performance of each model is evaluated by observing the accuracy of predictions. For the given feedbacks from the test set, we compare the rating prediction $\hat{r}_{i,j,s}$ with its actual rating $r_{i,j,s}$. Then, as evaluation indicators, we employ MAE and RMSE which are calculated as follows:

$$MAE = \frac{1}{N}\sum_{i,j,s}(|r_{i,j,s} - \hat{r}_{i,j,s}|)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i,j,s}(r_{i,j,s} - \hat{r}_{i,j,s})^2},$$

where N denotes the number of feedbacks in the test set, and $|\cdot|$ denotes the absolute value. In general, RMSE is more sensitive than MAE for the large error of prediction. For each of the training sets, every method is trained until the objective function converges.

B. Results of Evaluation

In this subsection, we show the results of the evaluation. Then, in the figures and tables, we abbreviate Doc2Vec to D2V.

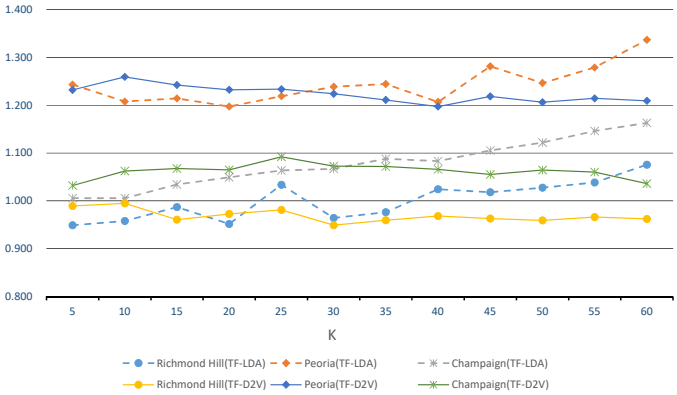

Fig. 3. RMSE of TF-Doc2Vec with K fixed from 5 to 60

First, to compare three TF methods, i.e., the basic TF, TF-LDA, and TF-Doc2Vec, we show the average RMSE for all values $K = 5, 10, 15, \cdots, 60$ and the p-value of the t-test in TABLE II. Recall that K denotes the number of overall features, also the dimension of U_i, V_j, and C_s. By these results, TF-Doc2Vec provides significantly better performance than others.

Next, Fig. 3 shows the RMSE values of TF-Doc2Vec and TF-LDA with K changed from 5 to 60. In the case of TF-Doc2Vec, the value of RMSE is stable, and the best performance with $K = 30$ for Richmond Hill and $K = 40$ for Peoria. Even though, when $K = 5$, Champaign has the best performance. In the case of TF-LDA, RMSE increases directly with K.

In order to achieve the best performance for other methods, K should not be assigned too small or too large values. So, we set K to 10, 20, 30, 40 and 50 to conduct a following detailed evaluation of the performance in rating prediction. TABLE III and TABLE IV summarize the results by six methods, with the best performance emphasized in boldface for each dataset. The improvement of the proposed methods (difference from other methods) is presented in the last five columns for each dataset. The value out of (resp. in) parentheses is of TF-Doc2Vec (resp. TF-LDA). The improvement of TF-Doc2Vec (resp. TF-LDA) is calculated by $(A - B)/A$ where A is the value by other method and B is the value by TF-Doc2Vec (resp. TF-LDA). Also, among TF methods, we conducted the t-test for each city and each K value. The improvement emphasized in boldface represents significant.

First, TF-Doc2Vec outperforms MF-Doc2Vec significantly in three cities. Similarly, comparing basic TF (resp. TF-LDA) with basic MF (resp. MF-LDA), the performance of

TABLE II
AVERAGE RMSE AND t-TEST FOR THREE CITIES AND ALL

	Average RMSE			p-**value by** t-**test**		
	TF	TF-LDA	TF-D2V	TF-LDA vs TF	TF-D2V vs TF	TF-D2V vs TF-LDA
Peoria	1.331222613	1.242816325	**1.223260637**	1.26335E-20	1.81139E-49	0.001609824
RichmondHill	1.030129732	1.000267133	**0.96881231**	9.41793E-06	5.53647E-38	2.35257E-09
Champaign	1.149275704	1.077718743	**1.062079227**	4.62652E-18	3.81717E-48	0.005940103
Average	1.17020935	1.106934067	**1.084717391**	6.85865E-39	3.2805E-122	2.08312E-11

TABLE III
THE PERFORMANCE IN TERMS OF RMSE OF SIX METHODS ON ALL DATASETS

	RMSE						**improvement by** TF-Doc2Vec (TF-LDA) (%)				
Dataset	MF	MF-LDA	MF-D2V	TF	TF-LDA	TF-D2V	vs MF	vs MF-LDA	vs MF-D2V	vs TF	vs TF-LDA (TF-D2V)
K=10											
Richmond Hill	1.0561	0.9892	1.1124	1.0661	**0.958**	0.9949	5.79 (9.29)	-0.58 (3.15)	10.56 (13.88)	**6.68** (**10.14**)	-3.85 (**3.71**)
Peoria	1.3167	1.3064	1.4613	1.358	**1.2076**	1.2592	4.37 (8.29)	3.61(7.56)	13.83 (17.36)	**7.28** (**11.08**)	-4.27 (**4.10**)
Champaign	1.1724	1.1587	1.3568	1.1622	**1.0056**	1.0625	9.37 (14.23)	8.30 (13.21)	21.69 (25.88)	**8.58** (**13.47**)	-5.66 (**5.36**)
Average	1.1817	1.1514	1.3102	1.1954	**1.0571**	1.1055	6.51 (10.54)	3.78 (8.19)	15.36 (19.32)	**7.51** (**11.57**)	-4.59 (**4.38**)
K=20											
Richmond Hill	1.0603	1.0244	1.2526	1.0214	**0.9516**	0.9728	8.25 (10.25)	5.04 (7.11)	22.34 (24.03)	4.76 (**6.83**)	-2.23 (2.18)
Peoria	1.358	1.5692	1.4774	1.3404	**1.1971**	1.2323	9.26 (11.85)	21.47 (23.71)	16.59 (18.97)	**8.06** (**10.69**)	-2.93 (2.86)
Champaign	1.2144	1.2425	1.38	1.1638	**1.0491**	1.0648	12.32 (13.61)	14.30 (15.57)	22.84 (23.98)	**8.51** (**9.86**)	-1.50 (1.47)
Average	1.2109	1.2787	1.37	1.1752	**1.0659**	1.09	9.94 (11.97)	13.60 (16.64)	20.59 (22.20)	**7.11** (**9.30**)	-2.22 (**2.21**)
K=30											
Richmond Hill	1.1097	1.0594	1.442	1.0209	0.9642	**0.9491**	14.47 (13.11)	10.41 (8.99)	34.18 (33.13)	**7.03** (**5.55**)	1.57 (-1.59)
Peoria	1.375	1.3614	1.5004	1.2968	1.2386	**1.2238**	11.00 (9.92)	10.11 (9.02)	18.44 (17.45)	**5.63** (**4.49**)	1.19 (-1.21)
Champaign	1.2803	1.2146	1.3864	1.1461	**1.0668**	1.0726	16.22 (16.68)	11.69 (12.17)	22.63 (23.06)	**6.41** (**6.92**)	-0.54 (0.54)
Average	1.255	1.2118	1.4429	1.1546	1.0899	**1.0818**	13.90 (13.16)	10.74 (10.06)	25.08 (24.46)	**6.36** (**5.60**)	0.74 (-0.75)
K=40											
Richmond Hill	1.2219	1.2032	1.6599	1.0175	1.0242	**0.9682**	20.77 (16.18)	19.53 (14.88)	41.67 (38.30)	4.85 (-0.66)	**5.47** (-5.78)
Peoria	1.3982	1.3757	1.5244	1.2742	1.2069	**1.1972**	14.37 (13.68)	12.97 (12.27)	21.46 (20.83)	**6.04** (**5.28**)	0.80 (-0.81)
Champaign	1.3045	1.2528	1.4522	1.149	1.0832	**1.0657**	18.31 (16.96)	14.93 (13.54)	26.61 (25.41)	**7.24** (**5.73**)	1.62 (-1.64)
Average	1.3082	1.2772	1.5455	1.1469	1.1048	**1.077**	17.82 (15.55)	15.81 (13.50)	29.91 (28.52)	**6.04** (**3.67**)	**2.63** (-2.58)
K=50											
Richmond Hill	1.2946	1.1304	1.8673	1.0168	1.0275	**0.9591**	25.91 (20.63)	15.15 (9.10)	48.64 (44.97)	**5.68** (-1.05)	**6.66** (-7.13)
Peoria	1.4875	1.4365	1.6067	1.3207	1.2464	**1.2061**	18.92 (16.21)	16.04 (13.23)	24.94 (22.42)	**8.68** (**5.63**)	**3.23** (-3.34)
Champaign	1.3598	1.28	1.5225	1.1341	1.122	**1.0645**	21.72 (17.49)	16.83 (12.34)	30.08 (26.31)	**6.14** (1.07)	5.12 (-5.40)
Average	1.3806	1.2823	1.6655	1.1572	1.132	**1.0766**	22.18 (18.01)	16.01 (11.72)	34.55 (32.03)	**6.83** (2.18)	**5.00** (-5.15)
Average for all	1.3146	1.2571	1.5513	1.1529	1.1089	**1.0785**	17.97 (15.65)	16.01 (11.79)	29.85 (28.52)	**6.41** (**3.82**)	2.79 (-2.82)

corresponding TF methods have increased to some extent. It indicates that the user's rating would be affected certainly by seasons.

TF-Doc2Vec shows the best performance in terms of RMSE and MAE in most of the cases that $K \geq 30$. Comparing with MF methods, the improvement of TF-Doc2Vec is more than 10% in both RMSE and MAE. Additionally, TF-Doc2Vec is significantly different from basic TF in almost all cases.

TF-LDA presents a better result than TF-Doc2Vec in the cases $K \leq 20$. But, the performance of TF-LDA would become bad along with the increment of K. When $K = 50$, the improvement of RMSE (resp. MAE) of TF-Doc2Vec against TF-LDA have increased up to 6.66% (resp. 7.17%). The main reason is that much bigger K makes the topics derived from reviews by using LDA not clear, further affects the performance. As for using Doc2Vec model to extract feature, it assigns feature vectors to each user which can represent the user's preferences and features. Different from LDA, Doc2Vec does not suffer the effect of varieties of K. So the performance of TF-Doc2Vec is almost stable.

VII. CONCLUSION

In this paper, we propose two new TF models TF-Doc2Vec and TF-LDA. From the given textual reviews, TF-Doc2Vec (resp. TF-LDA) extracts feature of users by Doc2Vec (resp. LDA), and these feature vectors affect the learning process of TF. In the evaluation, we conduct a series of experiments utilizing three cities information about restaurants from YELP challenge dataset. According to the comparison, the RMSE of rating prediction by TF-Doc2Vec and TF-LDA improves 5%-13% significantly comparing to the basic TF methods on average of all three cities by $K = 5, \cdots, 60$. Comparing with TF-LDA, the improvement of TF-Doc2Vec is determined by the size of the latent factor.

In the future, we plan to analyze the user's sentiment factors from reviews. Sometimes, some users tend to write reviews that contain positive sentiment or negative sentiment. Also, user's ratings might be affected by such sentiment bias to some extent.

REFERENCES

[1] G.-S. Fang, S. Kamei, and S. Fujita, "Rating prediction with topic gradient descent method for matrix factorization in recommendation,"

TABLE IV
THE PERFORMANCE IN TERM OF MAE OF SIX METHODS ON ALL DATASETS

	MAE						improvement by TF-Doc2Vec (TF-LDA) (%)				
Dataset	MF	MF-LDA	MF-D2V	TF	TF-LDA	TF-D2V	vs MF	vs MF-LDA	vs MF-D2V	vs TF	vs TF-LDA (TF-D2V)
K=10											
Richmond Hill	0.8323	0.7737	0.8645	0.8387	**0.7518**	0.7829	5.94 (9.67)	-1.19 (2.83)	9.44 (13.04)	**6.65** (**10.36**)	-4.14 (**3.97**)
Peoria	1.0397	1.0573	1.1808	1.0691	**0.9543**	0.9882	4.95 (8.21)	6.54 (9.74)	16.31 (19.18)	**7.57** (**10.74**)	-3.55 (**3.43**)
Champaign	0.92	0.9412	1.0825	0.916	**0.7909**	0.8389	8.82 (14.03)	10.87 (15.97)	22.50 (26.94)	**8.42** (**13.66**)	-6.07 (**5.72**)
Average	0.9307	0.9241	1.0426	0.9413	**0.8323**	0.87	6.57 (10.57)	5.41 (9.93)	16.08 (20.17)	**7.55** (**11.58**)	-4.59 (**4.33**)
K=20											
Richmond Hill	0.8398	0.8068	0.9629	0.7965	**0.7486**	0.765	8.91 (10.86)	5.18 (7.21)	20.55 (22.26)	3.95 (6.01)	-2.19 (2.14)
Peoria	1.071	1.2277	1.1848	1.0545	**0.9397**	0.978	8.68 (12.26)	20.34 (23.46)	17.45 (20.69)	**7.25** (**10.89**)	-4.07 (**3.92**)
Champaign	0.9597	1.012	1.1092	0.9213	**0.8278**	0.8395	12.52 (13.74)	17.05 (18.20)	24.31 (25.37)	**8.88** (**10.15**)	-1.41 (1.39)
Average	0.9568	1.0155	1.0853	0.9241	**0.8387**	0.8608	10.04 (12.34)	14.19 (17.41)	20.77 (22.72)	**6.70** (**9.24**)	-2.56 (**2.57**)
K=30											
Richmond Hill	0.8691	0.8369	1.1071	0.8024	0.7618	**0.7481**	13.92 (12.35)	10.62 (8.97)	32.43 (31.19)	**6.76** (**5.06**)	**1.80** (-1.83)
Peoria	1.0745	1.1029	1.1977	1.0148	0.9719	**0.9663**	10.08 (9.55)	12.39 (11.88)	19.32 (18.85)	**4.78** (**4.23**)	0.58 (-0.58)
Champaign	1.0068	0.9846	1.1052	0.9005	**0.8398**	0.8464	15.93 (16.59)	14.03 (14.71)	23.41 (24.01)	**6.00** (**6.74**)	-0.79 (0.78)
Average	0.9835	0.9748	1.1367	0.9059	0.8578	**0.8536**	13.31 (12.78)	12.35 (12.00)	25.05 (24.54)	**5.85** (**5.31**)	0.53 (-0.49)
K=40											
Richmond Hill	0.9601	0.9352	1.2581	0.8003	0.8054	**0.7643**	20.39 (16.11)	18.27 (13.88)	39.25 (35.98)	4.49 (-0.64)	**5.10** (-5.38)
Peoria	1.09	1.1138	1.1979	1.0017	0.9512	**0.9463**	13.18 (12.73)	15.03 (14.60)	21.00 (20.59)	**5.53** (**5.04**)	0.52 (-0.52)
Champaign	1.0227	1.0098	1.1435	0.9035	0.8543	**0.8448**	17.40 (16.47)	16.34 (15.40)	26.12 (25.29)	**6.50** (**5.45**)	1.11 (-1.12)
Average	1.0243	1.0196	1.1998	0.9018	0.8703	**0.8518**	16.99 (15.03)	16.55 (14.64)	28.79 (27.46)	**5.51** (**3.49**)	**2.24** (-2.17)
K=50											
Richmond Hill	1.0129	0.8865	1.409	0.805	0.8092	**0.7512**	25.84 (20.11)	15.27 (8.72)	46.69 (42.57)	**6.69** (-0.52)	**7.17** (-7.72)
Peoria	1.1535	1.1613	1.2547	1.0487	1.0034	**0.9626**	16.55 (13.01)	17.11 (13.60)	23.28 (20.03)	**8.21** (**4.32**)	**4.07** (-4.24)
Champaign	1.0761	1.0288	1.2028	0.8906	0.8937	**0.8465**	21.34 (16.95)	17.73 (13.13)	29.63 (25.70)	**4.95** (-0.35)	5.28 (-5.58)
Average	1.0808	1.0255	1.2888	0.9148	0.9021	**0.8534**	21.24 (16.53)	16.70 (12.03)	33.2 (30.00)	**6.62** (1.39)	**5.51** (-5.71)
Average for all	1.0808	1.0066	1.2084	0.9075	0.8767	**0.8538**	17.18 (18.88)	15.20 (12.90)	29.01 (27.45)	**5.99** (**3.39**)	2.76 (-2.68)

INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, vol. 8, no. 12, pp. 469–476, 2017.

[2] W.-T. Kuo, Y.-C. Wang, R. T.-H. Tsai, and J. Y. jen Hsu, "Contextual restaurant recommendation utilizing implicit feedback," in *Proceedings of the 24th Wireless and Optical Communication Conference (WOCC)*, 2015.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] S. Setiowati, T. B. Adji, and I. Ardiyanto, "Context-based awareness in location recommendation system to enhance recommendation quality: A review," in *Proceedings of International Conference on Information and Communications Technology (ICOIACT)*, 2018, pp. 90–95.

[5] J. Zeng, F. Li, H. Liu, J. Wen, and S. Hirokawa, "A restaurant recommender system based on user preference and location in mobile environment," in *Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2016, pp. 55–60.

[6] J. Bao, Y. Zheng, and M. F. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," in *Proceedings of the 20th international conference on advances in geographic information systems*, 2012, pp. 199–208.

[7] M. A. Habib, M. A. Rakib, and M. A. Hasan, "Location, time, and preference aware restaurant recommendation method," in *Proceedings of the 19th International Conference on Computer and Information Technology (ICCIT)*, 2016, pp. 315–320.

[8] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[9] P. Symeonidis and A. Zioupos, *Matrix and Tensor Factorization Techniques for Recommender Systems*. Springer, 2016.

[10] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, 2008, pp. 1257–1264.

[11] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 79–86.

[12] L. Yao, Q. Z. Sheng, Y. Qin, X. Wang, A. Shemshadi, and Q. He, "Context-aware point-of-interest recommendation using tensor factorization with social regularization," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 1007–1010.

[13] 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*, 2013, pp. 165—-172.

[14] Y. Bao, Y. Bao, and J. Zhang, "Topicmf: Simultaneously exploiting ratings and reviews for recommendation," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 2–8.

[15] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 2, 2011.

[16] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1188–1196.