International Journal of Networking and Computing – `www.ijnc.org`, ISSN 2185-2847
Volume 10, Number 2, pages 111–126, July 2020

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

Yang Sun
Dept. of Information Engineering, Graduate School of Engineering, Hiroshima University,
1-4-1 Kagamiyama, Higashi Hiroshima, Hiroshima, 739-8527 JAPAN

Guan-Shen Fang
National Institute of Technology, Tsuyama College,
624-1, Numa, Tsuyama-City, Okayama, 708-8509 JAPAN

Sayaka Kamei
Graduate School of Advanced Science and Engineering, Hiroshima University,
1-7-1 Kagamiyama, Higashi Hiroshima, Hiroshima, 739-8521 JAPAN

Received: February 12, 2020
Revised: May 4, 2020
Accepted: June 1, 2020
Communicated by Kouzou Ohara

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. Therefore, 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 [6, 14] have already approved the drawback of the ignorance of textual reviews would bring mediocre performance for rating prediction.

In this work, we propose a framework TF+ for rating prediction models based on Tensor Factorization (TF) which is an extended version of Matrix Factorization (MF) by adding another dimension. We consider seasonal context as the additional dimension. Firstly, in our framework, each of the reviews is characterized by a numeric feature vector. Secondly, it uses TF which is trained by the proposed first-order gradient descent method for TF named Feature Vector Gradient Descent (FVGD). For the training of TF in TF+, FVGD decides the learning rates based on the feature vectors of reviews. In our evaluation, we use pre-processed data of five cities in YELP challenge dataset, and apply one of LDA, Doc2Vec and SCDV to get numeric feature vectors of reviews. We conduct experimental comparisons, and the results show that methods by TF+ improve the performance significantly as compared to the basic TF model.

Keywords: Rating prediction, Tensor factorization, Doc2Vec, LDA, SCDV, Recommendation system

1 Introduction

Nowadays, recommender systems are an essential part of online services and social networks [20]. The systems would give recommendations to users by using feedback from them. 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 relationship with ratings to some extent. From reviews, we can roughly understand why users give such ratings for the 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. Despite the high accuracy that the CF method can offer, data sparsity still causes a big effect for first-used users that contain little information.

Among all the CF methods, Matrix Factorization (MF) is an ideal method to predict ratings [13]. However, recent research pointed out that the ignorance of the context is the major shortcoming of basic MF. Setiowati *et al.* [23] published a survey paper and pointed out that the context-aware approach is being developed by utilizing contextual information to produce more precise recommendations according to user's preferences. The results of some studies [1, 8, 14, 28] have shown that the implementation of context-awareness on a recommendation system has given better results for personalized recommendations rather than the ones without it. Additionally, recent research [4,6,14] pointed out that the ignorance of the reviews is the major shortcoming of basic MF and brings it mediocre performance.

For context-awareness on model-based recommendation systems, Tensor Factorization (TF) [12] is used by adding a context dimension to MF. Based on TF, many researchers proposed context-aware recommender systems, e.g., [24], [9], [26]. Frolov et al. showed a good survey in [7]. However, they also ignore the textual reviews or are inefficient even if they consider the textual reviews. If they consider the textual reviews, they could extract some particular information elements (e.g., the user's preference, the user's attributes, and the specific contexts and aspects) from the textual reviews by text analysis and mining techniques, and combine them as an additional dimension for each element into TF model [4].

In this paper, in order to offer a high quality of recommendation, we propose a new framework TF+ of methods to predict the rating on the user's purposeful POI for the recommendation by considering seasonal context factors and reviews by extending TF [12]. Various contexts appear in the information attached to the feedback and in the review text itself, such as the distance and the temperature of that time. However, in this paper, we will consider the season from the posting date, which is easy to obtain from the information attached to each feedback (not textual reviews). Although it is difficult to estimate the actual day and time when the reviewer went to the POI from the posting date, by expanding the time range to the season, we consider that the season when he/she went to the POI can be estimated from the posting date.

In TF+, we use textual reviews for the training of such TF. In methods of TF+, firstly, we set up to construct feature vectors of reviews by a method to get distributed representations of documents, e.g., LDA [3], Doc2Vec [15], SCDV [17]. We assume that the 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 our framework, against each visited POI, each of the users would be assigned a unique feature vector with numeric values. After that, the method by TF+ learns the feature vectors of users by the proposed first-order gradient descent method, called Feature Vector Gradient Descent (FVGD). It decides the learning rates based on the feature vectors of reviews.

In our evaluation, we conduct experiments using data about restaurants in five cities in YELP challenge dataset. We compare the performance of the following eight methods: basic MF method, three MF-based methods with FVGD (i.e., MF-LDA [6], MF-Doc2Vec and MF-SCDV), basic TF method, three methods by our framework (i.e., TF-LDA, TF-Doc2Vec and TF-SCDV). 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 contributions of this paper are as follows:

- We use a more complex TF model framework TF+ 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 relationship with the latent factor to some extent. To consider textual review contents, we propose a new gradient descent method FVGD to train the model by TF+.
- In the evaluation, we compare the performance of our methods with other five approaches in rating prediction. The results obviously show that the performance of context-aware TF-based methods is better than MF-based methods respectively. Also, proposed methods by TF+ 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 our framework. Section VI represents the experiment to compare eight methods including proposed methods. Finally, section VII concludes the paper with future work.

2 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 [25]. Salakhutdinov *et al.* [22] proposed 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 method, 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 their experimental result has shown that their method improved upon non-contextual basic MF up to 30% in terms of the MAE. Yao *et al.* [27] 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 with the MF model, [16] and [2] proposed methods with the transformation of the 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.* [6] proposed a novel method MF-LDA of rating prediction which uses both the ratings and reviews, including a 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. While MF-LDA is without the complicated transformation, it provides better performance compared with methods in [16] and [2]. We will explain more details about this in section 4.

3 Preliminaries

3.1 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. 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 $r_{i,j,s}$. We assume that, if $r_{i,j,s}$ exists in the set of feedbacks, 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[1] $\hat{r}_{i,j,s}$ for a given user u_i and a given item v_j under season c_s.

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

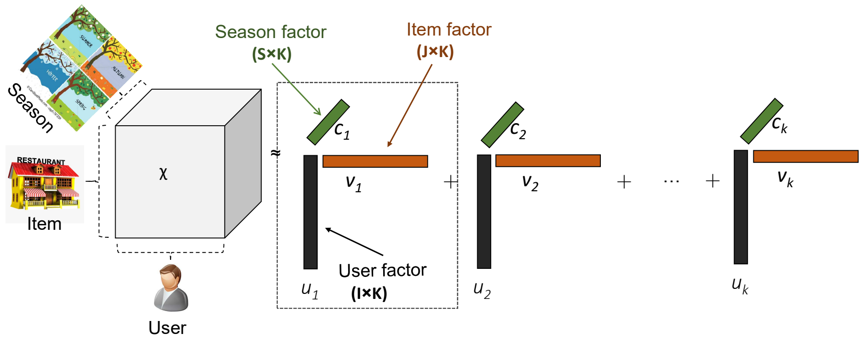


Figure 1: The basic TF (CP tensor decomposition).

3.2 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 [13] is also an influential method to predict the missing ratings based on the 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 user u_i. The elements of the vector in the 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 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 objective of biased MF (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:

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

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

3.3 Tensor Factorization for Recommendation

The CP-tensor decomposition is one of TF algorithms [5], [27]. We call it basic TF. 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 Figure 1, we show a three-dimensional space, i.e. a tensor χ, as a predicted rating model for the recommendation system. It maps users, items, and contexts into a joint latent factor space with K dimensions where K is arbitrarily predefined. Unlike MF in section 3.2, let C be an additional 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 basic TF is also to minimize the following function of regularized squared error:

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

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 squared error would be converged with a proper learning rate.

3.4 The methods of natural languages processing

In our framework, to get distributed representations of documents (i.e., reviews), we use one of the following Natural Languages Processing (NLP) methods: Latent Dirichlet Allocation (LDA) [3], Document to Vector (Doc2Vec) [15], Sparse Composite Document Vectors (SCDV) [17].

LDA is a probabilistic generative latent topic model of a set of semantic documents. Its idea is that each latent topic is characterized by a distribution over words, and a document is a random mixture over such topics. It models a document as a distribution of multiple latent topics, thus we can regard the topic distribution of a document as its feature vector. For LDA, if we set the number of topics to too large, the topics in each document cannot be distinguished well in the topic distribution.

Doc2Vec is also one of the widely used techniques for document vector representation in recent years. It is inspired by the word-level vector representation method Word2Vec [18] using neural networks. By using vector of words from Word2Vec, Doc2Vec further trains the vector representation for documents considering the sequence of words. Therefore, different from the traditional method of Bag-of-words and the way of summing the word vectors by Word2Vec in each document, it can characterize the semantic features of component words of documents. We use the Distributed Memory Model of Paragraph Vectors (PV-DM) as the training method for Doc2Vec in this paper.

SCDV is a new document vector representation method that can capture some special features of words like synonyms and polysemy. Through some extensive experiments in [17], the SCDV significantly outperforms the Doc2Vec method in the document classification tasks. By using Word2Vec and Gaussian Mixture Models (GMM) [19] which is a soft clustering technique, SCDV combines word embedding models and a latent topic model. That is, it combines syntax and semantics.

In Doc2Vec and SCDV, the characterization of words (i.e., their features in syntax and semantics) in each document is not changed largely by the number of dimensions of the document vectors.

4 Existing Method

In this section, we explain an existing method MF-LDA proposed by Fang *et al.* [6]. 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.

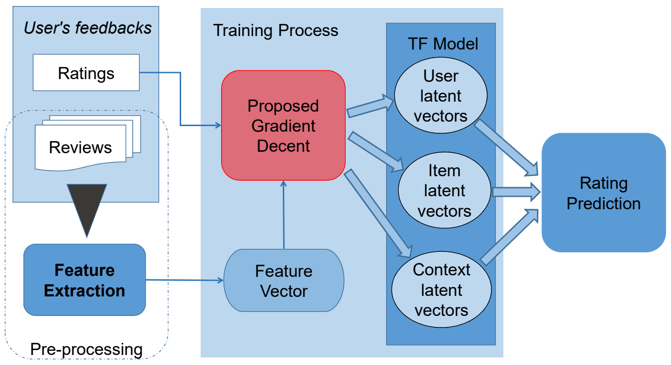


Figure 2: The construction of TF+.

In MF-LDA, they derive topic distribution from user's reviews via LDA. MF-LDA binds the latent topics to latent factors via the training process of MF instead of the complicated transformation as in [16] and [2].

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 $\theta_{i,j}^k$ 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 $\theta_{i,j}^k$ 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

$$U_i \leftarrow U_i - \gamma_M H_{i,j}^{\theta} \cdot gU_i,$$
$$V_j \leftarrow V_j - \gamma_M H_{i,j}^{\theta} \cdot gV_j,$$

where γ_M is a pre-defined constant for MF model, and $H_{i,j}^{\theta}$ is a $K \times K$ diagonal matrix such that the k-th diagonal element is $\theta_{i,j}^k$. $H_{i,j}^{\theta}$ 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 a given item v_j, we calculate the rating prediction $\hat{r}_{i,j}$ using Equation (1).

5 Proposed Method

In this section, we propose a framework TF+ of methods to predict missing ratings, based on the basic TF. The structure of TF+ is shown in Figure 2. In the method by TF+, first, one of the NLP methods in section 3.4 is used for feature extraction to get feature vectors of reviews. Then, if we use LDA (resp. Doc2Vec, SCDV), we call our method TF-LDA (resp. TF-Doc2Vec, TF-SCDV). After that, based on the feature vectors, methods by TF+ train TF models using a gradient descent method respectively. To this end, we propose a new first-order gradient descent method named Feature Vector Gradient Descent (FVGD) based on TGD [6].

From a set of user's reviews, the first task for our methods 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 one of NLP methods in section 3.4 to convert a document $d_{i,j,s}$ to a K-dimensional numeric vector $\delta_{i,j,s}$ called a feature vector. In each feature vector $\delta_{i,j,s}$, each element is weight $\delta_{i,j,s}^k$ corresponding to the latent factor $k \in \{1, \ldots, K\}$. The size of the corresponding latent factor K is determined by the dimension number of the feature vector. Then, we assume that $\delta^k_{i,j,s}$ represents how much interest user u_i shows to a latent factor k of item v_j in season c_s. That is, we assume that the feature vector $\delta_{i,j,s}$ represents the importance of the degree of features in the evaluation of u_i to v_j in c_s.

Next, we use a TF model to predict the missing rating values. The method by TF+ integrates the feature vectors of reviews and latent factors. So we propose FVGD to correlate them during the training of TF. 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

$$\begin{aligned} U_i &\leftarrow U_i - \gamma_T H^\delta_{i,j,s} \cdot gU_i, \\ V_j &\leftarrow V_j - \gamma_T H^\delta_{i,j,s} \cdot gV_j, \\ C_s &\leftarrow C_s - \gamma_T H^\delta_{i,j,s} \cdot gC_s, \end{aligned}$$

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 $\hat{r}_{i,j,s}$ using Equation (4).

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

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

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 (resp. MF-SCDV) when we use Doc2Vec (resp. SCDV). With the MF model trained by FVGD, for a given user u_i and a given item v_j, we calculate the rating prediction $\hat{r}_{i,j}$ using Equation (1).

6 Evaluation

6.1 Datasets and Implementation

We use *YELP challenge dataset*[2] which contains many types of business, such as supermarket, salon, event, and restaurant. Here, we only use data with the "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 the following constraints to guarantee the quality of feedback:

- 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 five cities whose data sparsity and whose average number of words in reviews are as large as possible. Then, from the dataset, we only independently utilize the feedbacks from following five cities: Champaign, Cleveland, Glendale, Peoria and Richmond Hill. For each review,

[2]https://www.yelp.com/

Table 1: The description of five cities of YELP used in experiments.

city	#feedbacks	avg.rating	#users	#items	sparsity	avg.words
Champaign	2824	3.7178	203	104	0.134	61.9
Cleveland	7637	3.9497	313	203	0.120	84.5
Glendale	4527	3.7018	346	211	0.062	62.8
Peoria	3116	3.6874	324	152	0.063	57.5
Richmond Hill	2336	3.4486	171	122	0.112	79.6

Table 2: Parameter setting.

Method	Parameter	Value
LDA	max_iter	500
	learning_offset	50
	learning_method	online (EM algorithm)
Doc2Vec	training model	PV-DM
	window	5
	min_count	5
	workers	5
	sample	1e-3
Word2Vec	model	skip-gram
	window	5
	min_count	5
	workers	5
	sample	1e-3

Table 3: Parameter setting for GMM.

Parameter		Value
convariance_type		tied
init_params		kmeans
max_iter		50
#clusters	Champaign	60
	Cleveland	100
	Glendale	60
	Peoria	20
	Richmond Hill	20
sparse threshold	Champaign	0.09
	Cleveland	0.06
	Glendale	0.07
	Peoria	0.05
	Richmond Hill	0.03

we discard the stop words, repeated words and punctuations, and do the stemming. With these pre-processes, Table 1 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[3], MF-Doc2Vec, MF-SCDV and basic TF with the same parameter values as TF-LDA, TF-Doc2Vec and TF-SCDV to guarantee fairness. For each of the training sets, we train each model until the squared error converges. To this end, we set the number of epochs of each model to 1000. 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 $[0, 1)$. To implement TF-LDA (resp. TF-Doc2Vec), we use LDA (resp. Doc2Vec) in sklearn (resp. gensim) library of Python. To implement TF-SCDV, we use Word2Vec and GMM in gensim and sklearn libraries respectively. We describe parameter settings for these methods changed from their default in Table 2. For GMM clustering, the number of clusters and the sparse threshold for each city are decided to their better values pre-experimentally as Table 3.

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 Mean Absolute

[3]We use basic MF as the base model of MF-LDA, while the original MF-LDA by Fang *et al.* uses biased MF. By our preliminary experiment, for our dataset, basic MF (resp. MF-LDA based on basic MF) outperforms biased MF (resp. the original MF-LDA).

Table 4: Average RMSE for five cities and all.

	TF	TF-LDA	TF-D2V	TF-SCDV
Champaign	1.1494	1.0777	1.0621	**1.0594**
Cleveland	1.0128	0.9600	0.9800	**0.9552**
Glendale	1.2253	**1.1348**	1.1491	1.1390
Peoria	1.3312	1.2428	1.2233	**1.2136**
Richmond Hill	1.0301	1.0003	0.9688	**0.9608**
Average	1.1498	1.0831	1.0767	**1.0656**

Table 5: t-test for five cities and all. The symbol * (resp. **) means that $p < 0.05$ ($p < 0.01$).

	TF-LDA vs TF	TF-D2V vs TF	TF-SCDV vs TF	TF-LDA vs TF-SCDV	TF-D2V vs TF-SCDV	TF-LDA vs TF-D2V
Champaign	4.63E-18 **	3.82E-48 **	3.46E-34 **	0.0024 **	0.5062	0.0059 **
Cleveland	2.23E-15 **	4.24E-25 **	1.18E-37 **	0.3605	6.17E-09 **	0.0028 **
Glendale	7.25E-24 **	1.36E-38 **	1.47E-41 **	0.5240	0.0173 *	0.0710
Peoria	1.26E-20 **	1.81E-49 **	1.45E-38 **	2.15E-05 **	0.0256 *	0.0016 **
Richmond Hill	9.42E-06 **	5.54E-38 **	0.0256 *	1.52E-09 **	0.0701	2.35E-09 **
Average	3.97E-73 **	2.52E-166 **	1.88E-158 **	1.52E-10 **	9.51E-09 **	0.0220 *

Error (MAE) and Root Mean Square Error (RMSE) which are calculated as follows:

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

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.

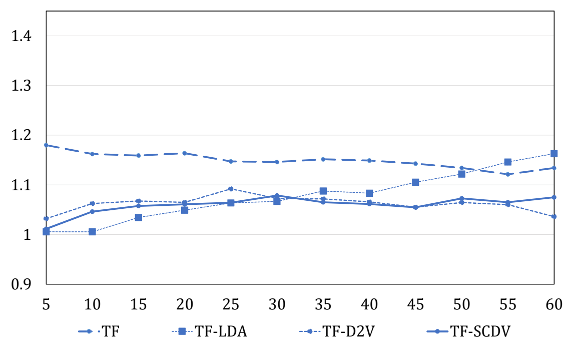
6.2 Results of Evaluation

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

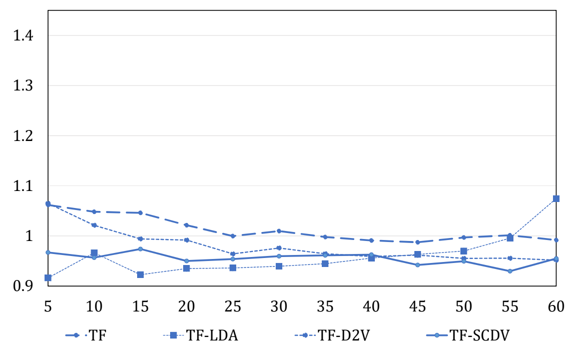
First, to compare four TF-based methods, i.e., the basic TF, TF-LDA, TF-Doc2Vec, and TF-SCDV, we show the average RMSE for all values $K = 5, 10, 15, \cdots, 60$ in Table 4 and the p-value of the t-test in Table 5. Recall that K denotes the number of overall features, also the dimension of U_i, V_j, and C_s. By these results, TF-SCDV provides significantly better performance than others. Additionally, methods of TF+ outperform basic TF significantly.

Next, to discuss the effect of dimension size K for each model, Figure 3 shows the RMSE values of basic TF, TF-Doc2Vec, TF-LDA and TF-SCDV with K changed from 5 to 60 for each city. In the cases of TF-Doc2Vec and TF-SCDV, the value of RMSE is almost stable and better than basic TF. In the case of TF-LDA, RMSE increases directly with K. However, in the case $K \leq 20$, TF-LDA outperforms other methods. To discuss this in detail, we calculate the perplexity score [10] and the coherence score [21] of LDA for each K by gensim. Perplexity is a measure of the ability of a model to generalize documents, and its score should be lower. On the other hand, coherence represents the average of the semantic similarities of words in topics, and its score should be higher. Figure 4 shows these scores with K changed from 5 to 60 for each city. From this result, the optimal number of K is smaller than 20 for any city. Note that, the dataset includes only reviews about restaurants. Thus, for TF-LDA, $K \leq 20$ is better. On the other hand, on TF-Doc2Vec and TF-SCDV, the difference in K has an insignificant effect.

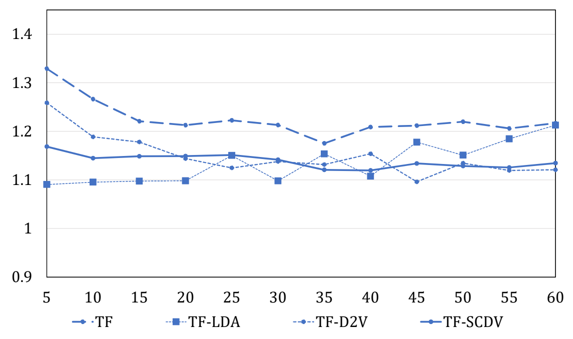
In the following, we set K to 10, 20, 30, 40 and 50 to conduct a detailed evaluation of the performance in rating prediction. Tables 6 and 8 summarize the results of RMSE and MAE respectively with the best performance emphasized in boldface for each dataset. Tables 7 and 9 summarize



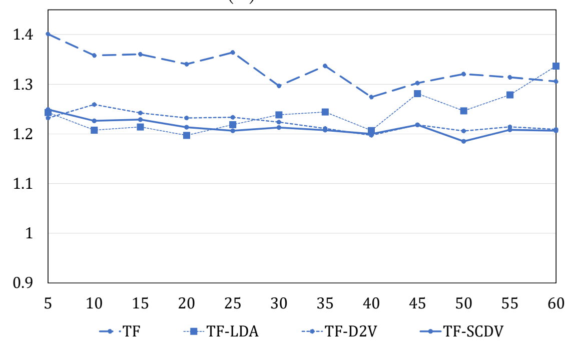
(a) Champaign.



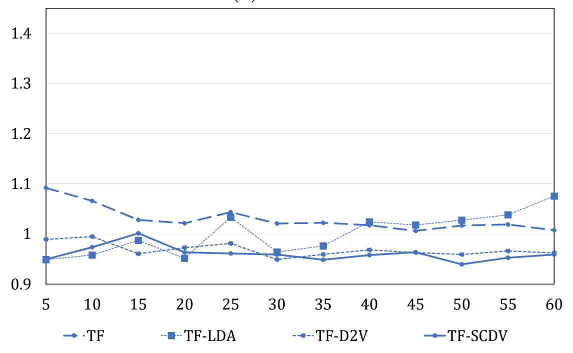
(b) Cleveland.



(c) Glendale.

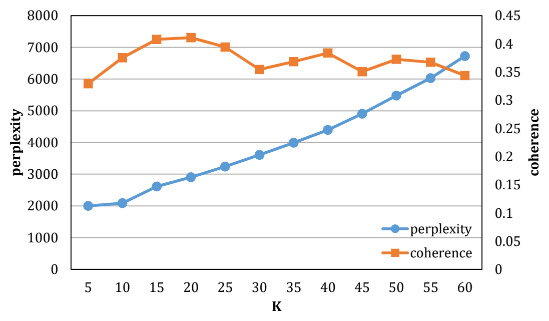


(d) Peoria.

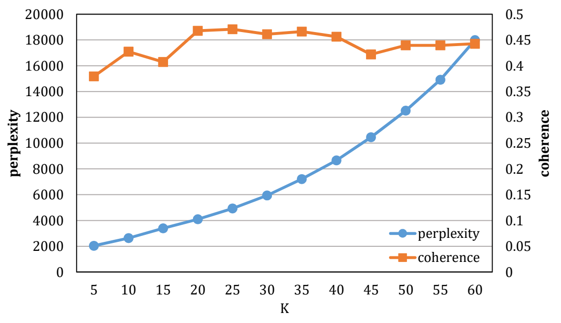


(e) Richmond Hill.

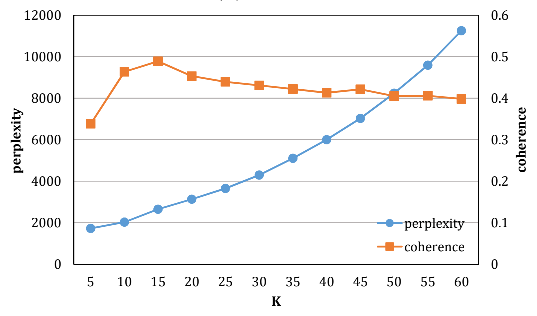
Figure 3: RMSE with K fixed from 5 to 60.



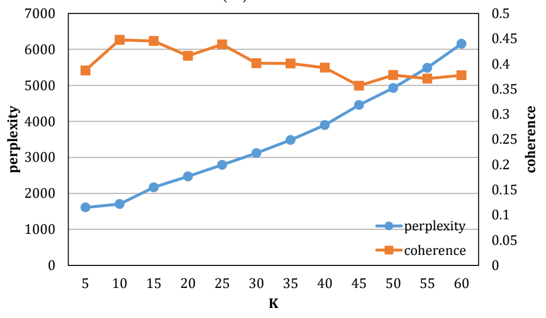
(a) Champaign.



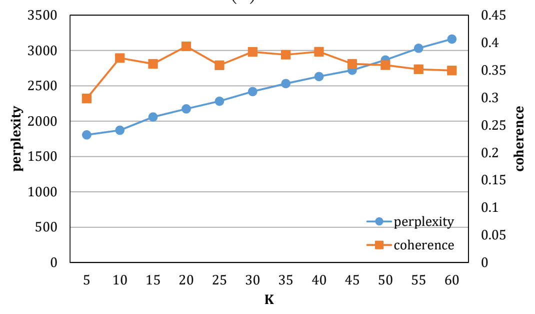
(b) Cleveland.



(c) Glendale.



(d) Peoria.



(e) Richmond Hill.

Figure 4: Perplexity and Coherence by LDA with K fixed from 5 to 60.

Table 6: The performance in terms of RMSE of eight methods on all datasets. The best performance is emphasized in boldface for each dataset.

K	Dataset	MF	MF-LDA	MF-D2V	MF-SCDV	TF	TF-LDA	TF-D2V	TF-SCDV
10	Champaign	1.1783	1.1041	1.1952	1.1067	1.1622	**1.0056**	1.0625	1.0463
	Cleveland	1.0720	0.9918	1.0652	0.9902	1.0482	0.9665	1.0214	**0.9568**
	Glendale	1.2364	1.1712	1.2458	1.1725	1.2663	**1.0955**	1.1888	1.1448
	Peoria	1.3254	1.2706	1.2943	1.2706	1.3580	**1.2076**	1.2592	1.2263
	Richmond Hill	1.0242	0.9862	1.0615	1.0017	1.0661	**0.9580**	0.9949	0.9736
	Average	1.1673	1.1048	1.1724	1.1083	1.1802	**1.0466**	1.1053	1.0696
20	Champaign	1.2210	1.1743	1.1894	1.1188	1.1638	**1.0491**	1.0648	1.0607
	Cleveland	1.1170	1.0541	1.1040	1.0479	1.0214	**0.9350**	0.9918	0.9500
	Glendale	1.2634	1.3046	1.2483	1.2264	1.2128	**1.0983**	1.1441	1.1491
	Peoria	1.3437	1.4289	1.3223	1.2702	1.3404	**1.1971**	1.2323	1.2136
	Richmond Hill	1.0824	1.1396	1.0655	1.0799	1.0214	**0.9516**	0.9728	0.9632
	Average	1.2055	1.2203	1.1859	1.1486	1.1520	**1.0462**	1.0811	1.0673
30	Champaign	1.2420	1.1679	1.2448	1.2013	1.1461	**1.0668**	1.0726	1.0787
	Cleveland	1.1310	1.0311	1.1474	1.0827	1.0097	**0.9396**	0.9759	0.9596
	Glendale	1.2869	1.2053	1.2962	1.2730	1.2131	**1.0981**	1.1380	1.1418
	Peoria	1.3657	1.3205	1.3670	1.3343	1.2968	1.2386	1.2238	**1.2130**
	Richmond Hill	1.1050	1.0680	1.1442	1.1179	1.0209	0.9642	**0.9491**	0.9588
	Average	1.2261	1.1586	1.2399	1.2018	1.1373	**1.0614**	1.0719	1.0704
40	Champaign	1.3019	1.1796	1.2998	1.2699	1.1490	1.0832	1.0657	**1.0615**
	Cleveland	1.1552	1.0612	1.1713	1.1356	0.9911	**0.9558**	0.9591	0.9628
	Glendale	1.3179	1.2263	1.3706	1.3566	1.2090	**1.1078**	1.1540	1.1197
	Peoria	1.4694	1.3362	1.4809	1.4907	1.2742	1.2069	**1.1972**	1.2000
	Richmond Hill	1.1986	1.1666	1.2044	1.2248	1.0175	1.0242	0.9682	**0.9579**
	Average	1.2886	1.1940	1.3054	1.2955	1.1282	1.0756	1.0688	**1.0604**
50	Champaign	1.3246	1.2072	1.3730	1.3431	1.1341	1.1220	**1.0645**	1.0728
	Cleveland	1.2089	1.0849	1.2309	1.2059	0.9970	0.9700	0.9552	**0.9495**
	Glendale	1.4545	1.2667	1.4715	1.4439	1.2198	1.1510	1.1345	**1.1289**
	Peoria	1.5248	1.7541	1.5625	1.6057	1.3207	1.2464	1.2061	**1.1852**
	Richmond Hill	1.3736	1.1608	1.3556	1.3490	1.0168	1.0275	0.9591	**0.9394**
	Average	1.3773	1.2947	1.3987	1.3895	1.1377	1.1034	1.0639	**1.0551**
	Average for all	1.2529	1.1945	1.2605	1.2288	1.1471	1.0667	1.0782	**1.0646**

the improvement of TF-SCDV (difference from other methods) for each dataset, and the best performance is also emphasized in boldface. The improvement is calculated by $(A - B)/A$ where A is the value by other methods and B is the value by TF-SCDV. Also, among TF-based methods, we conducted the t-test for each city and each K value. The improvement emphasized in boldface represents significant.

First, TF-SCDV outperforms MF-SCDV significantly in all cities while parameters are in the same settings. Similarly, comparing basic TF (resp. TF-LDA, TF-Doc2Vec) with basic MF (resp. MF-LDA, MF-Doc2Vec), the performance of the corresponding TF-based method have increased to some extent. It indicates that, in the feedbacks about restaurants, the user's rating would be affected certainly by seasons.

TF-SCDV shows the best performance in terms of RMSE and MAE in most of the cases that $K \geq 40$. Comparing with MF-based methods, the improvement of TF-SCDV is more than 10% in both RMSE and MAE. Additionally, TF-SCDV significantly outperforms basic TF in almost all cases. TF-LDA presents a better result than TF-SCDV in the cases $K \leq 20$, while the performance of TF-LDA would decrease along with the increment of K. When we consider the average performance, TF-SCDV and TF-LDA are similar and outperform TF-Doc2Vec.

Note that, in the case of MF-based methods, we got similar results, i.e., MF can be improved by NLP methods and MF-LDA is the best among MF-based methods.

Table 7: The improvement in terms of RMSE of TF-SCDV on all datasets (%). The improvement emphasized in boldface represents significant.

K	Dataset	vs MF	vs MF-LDA	vs MF-D2V	vs MF-SCDV	vs TF	vs TF-LDA	vs TF-D2V
10	Champaign	11.20	5.23	12.46	5.46	**9.97**	-4.05	1.53
	Cleveland	10.75	3.53	10.18	3.37	**8.72**	1.00	**6.32**
	Glendale	7.41	2.25	8.11	2.37	**9.60**	-4.50	3.70
	Peoria	7.48	3.49	5.26	3.49	**9.70**	-1.54	2.62
	Richmond Hill	4.94	1.28	8.28	2.81	**8.68**	-1.63	2.14
	Average	8.36	3.16	8.86	3.50	**9.33**	**-2.14**	**3.26**
20	Champaign	13.13	9.67	10.82	5.19	**8.85**	-1.10	0.38
	Cleveland	14.94	9.88	13.95	9.34	**6.99**	-1.60	4.21
	Glendale	9.04	11.92	7.95	6.30	**5.25**	**-4.63**	-0.44
	Peoria	9.69	15.07	8.22	4.46	**9.46**	-1.37	1.52
	Richmond Hill	11.01	15.48	9.60	10.80	**5.70**	-1.22	0.98
	Average	11.56	12.40	10.11	7.22	**7.25**	**-1.99**	1.33
30	Champaign	13.15	7.64	13.34	10.21	**5.88**	-1.12	-0.56
	Cleveland	15.15	6.93	16.36	11.37	**4.96**	**-2.13**	1.67
	Glendale	11.27	5.27	11.91	10.30	**5.87**	**-3.98**	-0.34
	Peoria	11.18	8.14	11.27	9.09	6.46	2.07	0.88
	Richmond Hill	13.23	10.22	16.20	14.23	6.08	0.55	-1.02
	Average	12.80	7.64	13.82	11.04	**5.85**	-0.92	0.13
40	Champaign	18.47	10.01	18.33	16.41	**7.61**	2.00	0.40
	Cleveland	16.65	9.27	17.80	15.22	**2.86**	-0.73	-0.39
	Glendale	15.04	8.69	18.31	17.46	**7.39**	-1.08	2.96
	Peoria	18.34	10.20	18.97	19.50	**5.83**	0.58	-0.23
	Richmond Hill	20.08	17.89	20.46	21.79	**5.86**	**6.47**	1.06
	Average	17.71	11.21	18.77	18.08	**5.91**	1.45	0.76
50	Champaign	19.01	11.13	21.87	20.13	**5.41**	4.38	-0.77
	Cleveland	21.46	12.48	22.87	21.27	**4.77**	2.12	0.60
	Glendale	22.39	10.88	23.28	21.82	**7.46**	1.92	0.50
	Peoria	22.27	32.43	24.15	26.19	**10.26**	**4.92**	1.73
	Richmond Hill	31.61	19.07	30.70	30.36	**7.62**	**8.57**	2.05
	Average	23.35	17.20	24.57	23.95	**7.10**	**4.38**	0.82
	Average for all	14.76	10.32	15.23	12.76	**7.09**	0.16	**1.26**

7 Conclusion

In this paper, we propose a new framework TF+ including TF-Doc2Vec, TF-LDA and TF-SCDV. From the given textual reviews, TF-Doc2Vec (resp. TF-LDA, TF-SCDV) extracts features of reviews by Doc2Vec (resp. LDA, SCDV), and these feature vectors affect the learning process of TF. In the evaluation, we conduct a series of experiments utilizing five cities information about restaurants from YELP challenge dataset. According to the comparison, the RMSE of rating prediction by methods in our framework TF+ improves 5.8–7.3% significantly comparing to the basic TF method on average of all five cities by $K = 5, \cdots, 60$. If $K \leq 20$, TF-LDA outperforms other methods, otherwise, TF-SCDV is better.

While we tried LDA, Doc2Vec and SCDV as the NLP methods for TF+ in this paper, in the future, we plan to identify the characteristics of NLP which are better or worse for our framework even if we apply other NLP methods. Additionally, while we tried basic MF and basic TF as base models for TF+, we also plan to try other new models.

References

[1] Jie Bao, Yu Zheng, and Mohamed 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*, pages 199–208, 2012.

Table 8: The performance in term of MAE of eight methods on all datasets. The best performance is emphasized in boldface for each dataset.

K	Dataset	MF	MF-LDA	MF-D2V	MF-SCDV	TF	TF-LDA	TF-D2V	TF-SCDV
10	Champaign	0.9322	0.8697	0.9406	0.8683	0.9160	**0.7909**	0.8389	0.8296
	Cleveland	0.8349	0.7700	0.8245	0.7713	0.8156	0.7462	0.7959	**0.7417**
	Glendale	0.9729	0.9169	0.9690	0.9175	0.9933	**0.8556**	0.9297	0.8999
	Peoria	1.0398	0.9995	1.0236	1.0039	1.0691	**0.9543**	0.9882	0.9641
	Richmond Hill	0.7993	0.7735	0.8303	0.7929	0.8387	**0.7518**	0.7829	0.7683
	Average	0.9158	0.8659	0.9176	0.8708	0.9265	**0.8198**	0.8671	0.8407
20	Champaign	0.9636	0.9321	0.9285	0.8843	0.9213	**0.8278**	0.8395	0.8349
	Cleveland	0.8725	0.8377	0.8621	0.8127	0.7969	**0.7288**	0.7718	0.7470
	Glendale	0.9900	1.0176	0.9745	0.9591	0.9589	**0.8539**	0.9005	0.8964
	Peoria	1.0652	1.1122	1.0452	1.0070	1.0545	**0.9397**	0.9780	0.9591
	Richmond Hill	0.8479	0.8845	0.8372	0.8400	0.7965	**0.7486**	0.7650	0.7534
	Average	0.9479	0.9568	0.9295	0.9006	0.9056	0.8198	0.8510	0.8382
30	Champaign	0.9805	0.9195	0.9819	0.9448	0.9005	**0.8398**	0.8464	0.8571
	Cleveland	0.8827	0.8089	0.8936	0.8421	0.7912	**0.7402**	0.7629	0.7476
	Glendale	1.0119	0.9485	1.0133	0.9980	0.9526	**0.8618**	0.8954	0.8914
	Peoria	1.0741	1.0473	1.0648	1.0291	1.0148	0.9719	0.9663	**0.9647**
	Richmond Hill	0.8625	0.8410	0.8915	0.8778	0.8024	0.7618	**0.7481**	0.7540
	Average	0.9624	0.9130	0.9690	0.9384	0.8923	**0.8351**	0.8438	0.8430
40	Champaign	1.0265	0.9388	1.0315	0.9911	0.9035	0.8543	0.8448	**0.8416**
	Cleveland	0.9038	0.8369	0.9113	0.8829	0.7777	**0.7498**	0.7520	0.7510
	Glendale	1.0321	0.9692	1.0758	1.0541	0.9553	**0.8753**	0.9054	**0.8753**
	Peoria	1.1503	1.0494	1.1517	1.1524	1.0017	0.9512	**0.9463**	0.9501
	Richmond Hill	0.9352	0.9180	0.9407	0.9522	0.8003	0.8054	0.7643	**0.7544**
	Average	1.0095	0.9425	1.0222	1.0065	0.8877	0.8472	0.8426	**0.8345**
50	Champaign	1.0428	0.9612	1.0844	1.0550	0.8906	0.8937	0.8465	**0.8459**
	Cleveland	0.9434	0.8492	0.9607	0.9368	0.7526	0.7974	**0.7327**	0.7394
	Glendale	1.1312	0.9966	1.1490	1.1217	0.9587	0.9069	0.8988	**0.8858**
	Peoria	1.1771	1.3297	1.2051	1.2321	1.0487	1.0034	0.9626	**0.9397**
	Richmond Hill	1.0703	0.9135	1.0563	1.0485	0.8050	0.8092	0.7512	**0.7342**
	Average	1.0729	1.0100	1.0911	1.0788	0.8962	0.8741	0.8407	**0.8290**
	Average for all	0.9817	0.9376	0.9859	0.959	0.9017	0.8392	0.8490	**0.8371**

[2] Yang Bao, Yang Bao, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 2–8, 2014.

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

[4] Li Chen, Guanliang Chen, and Feng Wang. Recommender systems based on user reviews:the state of the art. *User Modeling and User-Adapted Interaction*, (25):99–154, 2015.

[5] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data*, 5(2), 2011.

[6] Guan-Shen Fang, Sayaka Kamei, and Satoshi Fujita. Rating prediction with topic gradient descent method for matrix factorization in recommendation. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 8(12):469–476, 2017.

[7] Evgeny Frolov and Ivan Oseledets. Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(3):e1201, 2017.

[8] Md. Ahsan Habib, Md. Abdur Rakib, and Muhammad Abul Hasan. Location, time, and preference aware restaurant recommendation method. In *Proceedings of the 19th International Conference on Computer and Information Technology*, pages 315–320, 2016.

Table 9: The improvement in term of MAE of TF-SCDV on all datasets (%). The improvement emphasized in boldface represents significant.

K	Dataset	vs MF	vs MF-LDA	vs MF-D2V	vs MF-SCDV	vs TF	vs TF-LDA	vs TF-D2V
10	Champaign	11.00	4.61	11.80	4.46	**9.43**	-4.89	1.11
	Cleveland	11.17	3.67	10.04	3.83	**9.07**	0.60	**6.81**
	Glendale	7.51	1.86	7.13	1.92	**9.40**	**-5.18**	3.21
	Peoria	7.28	3.55	5.82	3.97	**9.82**	-1.03	2.44
	Richmond Hill	3.88	0.68	7.46	3.11	**8.39**	-2.19	1.86
	Average	8.17	2.87	8.45	3.46	**9.22**	**-2.54**	**3.09**
20	Champaign	13.36	10.43	10.09	5.58	**9.38**	-0.86	0.55
	Cleveland	14.38	10.83	13.36	8.09	**6.27**	-2.50	3.22
	Glendale	9.46	11.91	8.02	6.54	**6.52**	**-4.90**	0.46
	Peoria	9.96	13.76	8.24	4.76	**9.05**	-2.06	1.93
	Richmond Hill	11.15	14.82	10.01	10.31	**5.41**	-0.64	1.52
	Average	11.66	12.35	9.94	7.06	**7.32**	**-2.21**	1.53
30	Champaign	12.59	6.79	12.71	9.28	4.82	-2.06	-1.26
	Cleveland	15.31	7.58	16.34	11.23	**5.51**	**-1.00**	2.00
	Glendale	11.91	6.02	12.03	10.68	**6.42**	**-3.43**	0.45
	Peoria	10.19	7.88	9.40	6.25	4.94	0.74	0.17
	Richmond Hill	12.58	10.35	15.42	14.11	6.03	1.02	-0.79
	Average	12.51	7.72	13.18	10.31	**5.54**	-0.95	0.11
40	Champaign	18.01	10.35	18.41	15.08	**6.85**	1.49	0.38
	Cleveland	16.91	10.26	17.59	14.94	**3.44**	-0.17	0.13
	Glendale	15.19	9.69	18.63	16.96	**8.37**	0.00	3.32
	Peoria	17.40	9.47	17.50	17.55	**5.15**	0.12	-0.40
	Richmond Hill	19.33	17.82	19.81	20.77	5.74	**6.33**	1.30
	Average	17.37	11.52	18.39	17.06	**5.91**	1.55	0.95
50	Champaign	18.88	11.99	21.99	19.82	**5.02**	5.35	0.07
	Cleveland	21.62	12.93	23.04	21.07	**5.60**	3.00	1.33
	Glendale	21.70	11.12	22.91	21.03	**7.60**	2.33	1.45
	Peoria	20.17	29.33	22.02	23.73	**10.39**	**6.35**	2.38
	Richmond Hill	31.40	19.63	30.49	29.98	**8.80**	**9.27**	2.26
	Average	22.75	17.00	24.09	23.13	**7.35**	**5.13**	1.36
	Average for all	14.49	10.29	14.81	12.20	**7.07**	0.20	**1.41**

[9] Balázs Hidasi and Domonkos Tikk. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery*, 30(2):342–371, 2016.

[10] Matthew D. Hoffman, David M. Blei, and Francis Bach. Online learning for latent dirichlet allocation. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, pages 856–864, 2010.

[11] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM conference on Recommender systems*, pages 79–86, 2010.

[12] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[14] Wei-Ti Kuo, Yu-Chun Wang, Richard Tzong-Han Tsai, and Jane Yung jen Hsu. Contextual restaurant recommendation utilizing implicit feedback. In *Proceedings of the 24th Wireless and Optical Communication Conference*, 2015.

[15] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196, 2014.

[16] 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, 2013.

[17] Dheeraj Mekala, Vivek Gupta, Bhargavi Paranjape, and Harish Karnick. Scdv: Sparse composite document vectors using soft clustering over distributional representations. *arXiv preprint arXiv:1612.06778*, 2016.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119, 2013.

[19] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015.

[20] Francesco Ricci, Lior Rokach, and Bracha Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.

[21] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 399—-408, 2015.

[22] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1257–1264, 2008.

[23] Sulis Setiowati, Teguh Bharata Adji, and Igi Ardiyanto. Context-based awareness in location recommendation system to enhance recommendation quality: A review. In *Proceedings of International Conference on Information and Communications Technology*, pages 90–95, 2018.

[24] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. Tfmap: Optimizing map for top-n context-awarerecommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 155—-164, 2012.

[25] Panagiotis Symeonidis and Andreas Zioupos. *Matrix and Tensor Factorization Techniques for Recommender Systems*. Springer, 2016.

[26] Wenmin Wu, Jianli Zhao, Chunsheng Zhang, Fang Meng, Zeli Zhang, Yang Zhang, and Qiuxia Sun. Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding. *Knowle dge-Base d Systems*, (128):71–77, 2017.

[27] Lina Yao, Quan Z. Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi 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*, pages 1007–1010, 2015.

[28] Jun Zeng, Feng Li, Haiyang Liu, Junhao Wen, and Sachio 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*, pages 55–60, 2016.