A Two-Stage Item Recommendation Method using Probabilistic Ranking with Reconstructed Tensor Model

Noor Ifada*, Richi Nayak
Queensland University of Technology, Brisbane, Queensland 4000, Australia
noor.ifada@{if.trunojoyo.ac.id,qut.edu.au},r.nayak@qut.edu.au

Abstract. In a tag-based recommender system, the multi-dimensional <user, item, tag> correlation should be modeled effectively for finding quality recommendations. Recently, few researchers have used tensor models in recommendation to represent and analyze latent relationships inherent in multi-dimensions data. A common approach is to build the tensor model, decompose it and, then, directly use the reconstructed tensor to generate the recommendation based on the maximum values of tensor elements. In order to improve the accuracy and scalability, we propose an implementation of the n-mode block-striped (matrix) product for scalable tensor reconstruction and probabilistically ranking the candidate items generated from the reconstructed tensor. With testing on real-world datasets, we demonstrate that the proposed method outperforms the benchmarking methods in terms of recommendation accuracy and scalability.

***Keywords*:** tensor reconstruction, probabilistic ranking, item recommendation

1 Introduction

Web personalization has become a solution to overcome the problem of abundant information on the internet [1]. The personalized systems gather information about users to build user profiles, and tailor them to recommend items interesting to users. With the growing user-generated information on the web, the Social Tagging Systems (STS) have gained great popularity as they allow users to annotate items like websites (delicious.com) or artists (www.last.fm). These reusable and sharable tags reveal user interests implicitly [2], and serve as an additional source of information to build user profiles for recommender systems [3, 4]. The performance of tag-based recommender systems heavily relies on how the tag assignments, representing the <user, item, tag> correlation, have been exploited. The user profiles model should expose the latent relationship between users, items, and tags. With using two-dimensional modeling approach [5], the total interaction between the three dimensions may be lost, and this will result in poor recommendation accuracy [4, 6]. Since the tag assignment data is a multi-dimensional data, it becomes obvious that user profiles should be modeled with higher-order data models rather than projecting them into lower dimension approach.

Tensor modeling, a well-known approach to represent and analyze latent relationships inherent in multi-dimensions data [7], can be adapted in recommender systems. Researchers have used tensor models to recommend tags [4, 8-10] and items [4, 6, 11] to users. The nature of recommending item differs from the tag recommendation since item recommendation is generated based on the user information specified only while the later have more information about the subject to receive recommendation, i.e., the user-item combination [12]. There are two ways that a tensor model can be utilized in recommendation: (1) by decomposing the tensor model and inferring recommendations based on decomposition factors [8, 10]; and (2) by reconstructing the decomposed models and inferring recommendation from the reconstructed tensor [4, 6, 9, 11, 13]. As scalability is a common issue in the tensor model, existing studies propose to solve the problem within the decomposition process by implementing the memory efficient [14] and optimization criterion [10] methods.

The second type of approach, using reconstructed tensors, is a step further than the former. The model needs to be reconstructed, as an approximation of the initial tensor, to reveal the latent relationships between dimensions of the tensor model [6]. These latent relationships can form the basis of identifying new entries to be used as recommendations. Existing methods build the tensor model, decompose it using standard techniques and, then, directly use the reconstructed tensor to generate the recommendations based on the maximum values of tag assignments in each user-item combinations of tensor elements [4, 6, 11]. Those previous studies assume that tag assignment value in the reconstructed tensor represents the level of user

* Noor Ifada is currently on leave from University of Trunojoyo Madura, Indonesia

preference for an item based on the tag value. However, they ignore the user's tagging history that has been found most influencive in forming user likelihood to the recommended items in recommendation research [5]. We conjecture that ranking of these items BY utilizing the user's tagging history would improve the recommendation accuracy. Another problem with the existing approaches is scalability. Tensor reconstruction is an expensive process as all decomposed factors need to be multiplied to form an approximate tensor. We have not found the examples of reconstructing large size tensors (for instance, more than the size of $1000 \times 1000 \times 1000$ for a three-dimension tensor model) [4, 6, 8, 11, 13].

In this paper, we adopt tensor models for generating recommendation using the tensor reconstruction approach. Our focus is on improving scalability during the reconstruction step and improving recommendation accuracy after the model has been reconstructed. Approaching the complexity within the decomposition task is beyond the scope of this paper. We propose an item recommendation method that utilizes a memory efficient loop approach for scalable tensor reconstruction and a probabilistic ranking to improve the accuracy of recommendations generated from the reconstructed tensor. The memory efficient loop implements the n-mode block-striped (matrix) product to reconstruct the tensor by multiplying all decomposed elements. The probabilistic ranking calculates the probability of users to select candidate items generated from the reconstructed tensor using their tag preference list.

We evaluate the method with several variations implementing the two broad tensor decomposition technique families, Tucker (HOSVD, HOOI) and CP [7], on two real-world STS datasets. Extensive offline experiments have been conducted to find the effectiveness of the method with various sensitivity analysis over the benchmarking methods: conventional tensor-based method [4] and a state-of-the-art matrix-based method [5]. Empirical analysis shows that the proposed method is able to outperform the benchmarking methods in terms of recommendation accuracy and scalability.

2 A Two-Stage Tensor-based Recommendation

2.1 The Tag Assignment Data

Let $U = \{u_1, u_2, u_3, \dots, u_{|U|}\}$ be the set of all users, $I = \{i_1, i_2, i_3, \dots, i_{|I|}\}$ be the set of all items and $T = \{t_1, t_2, t_3, \dots, t_{|T|}\}$ be the set of all tags in the tag assignment data A. In STS, a vector of tag assignment, $a(u, i, t) \in A$ represents the tagging activity of user u for item i with tag t. For each tag assignment, possible value of v_A for $a(u, i, t)$ is {0, 1} where 1 indicates that $a(u, i, t)$ exists and 0 indicates otherwise. The post O denotes the set of all distinct user-item combinations in A, $dom(O) \subset dom(A)$, as a user can tag an item with multiple values. For each post, the possible value of v_O for $o(u, i)$ is {0, 1} where 1 indicates user u has tagged item i with a tag and 0 indicates user u has not tagged item i. The tag assignment data A is used in both generating the item recommendation candidates and ranking item recommendations, while the post data O is used for ranking item recommendations only.

2.2 High-Level Definition

Figure 1 illustrates the high-level definition of the proposed two-stage tensor-based recommendation method. The first stage is tensor reconstruction that generates candidate items which includes: initial third-order tensor constructed from the tag assignment data A, decomposed factors after factorization of the model, and the approximated tensor model generated with multiplications of the decomposed factors. The second stage is probabilistic ranking that generates the Top-N ranked list of item recommendations to users. This stage calculates the probability of a user to select a candidate item generated from the reconstructed tensor using his tag preference list and rank the items for a user according to their probability values.

2.3 Stage 1: Tensor Reconstruction

In this stage, we build an approximate tensor by using decomposed factors incorporating the latent relationships between the dimensions of users, items and tags. The reconstructed tensor will generate candidate items to be used in next stage for recommendation. From the tag assignment data A, an initial third order tensor $\mathcal{Y} \in \mathbb{R}^{U \times I \times T}$ is constructed where U, I, and T are the set of users, items and tags respectively. Each element of tensor is assigned with binary value v_A. The value will change to continuous value reflecting

the significance of triplet after the decomposition process. Only the non-zero values are used in tensor construction to allow the creation of large tensor models as well as handling the sparsity problem in the tag assignment data. A decomposition technique is applied to tensor $\mathcal{Y}$ in order to derive the latent relationships inherent in the dimensions. Two broad family of decomposition techniques are Tucker (including the Higher-Order SVD (HOSVD) and Higher-Order Orthogonal Iteration (HOOI) methods) and Can-decomp/Parafac (CP) [7]. CP can be considered as a special case of Tucker where the core tensor is diagonal [7]. In this section, we show the process of the third-order tensor decomposition using CP technique as illustrated in Figure 2. However, we also implemented the method with HOSVD and HOOI methods and empirically analyze the results as reported in the next section.

For the third-order tensor $\mathcal{Y} \in \mathbb{R}^{U \times I \times T}$, CP performs Singular Value Decomposition (SVD) on the matricized data [7] resulting three left singular matrices M_U, M_I, and M_T which correspond to each dimension of the tensor. By choosing size reduction of $j \in \{1, |U|\}$, , $k \in \{1, |I|\}$, and $l \in \{1, |T|\}$, the reduced factor matrices are obtained as $R_U \in \mathbb{R}^{U \times j}, R_I \in \mathbb{R}^{I \times k}, R_T \in \mathbb{R}^{T \times l}$. The diagonal core tensor $\mathcal{C}$ which defines the interaction between the users, items and tags [7] then can be calculated as:

$$\mathcal{C} = \mathcal{Y} \times_1 R_U{}' \times_2 R_I{}' \times_3 R_T{}' \tag{1}$$

where $\mathcal{C} \in \mathbb{R}^{j \times k \times l}$. The reconstructed tensor $\hat{\mathcal{Y}}$ is derived as:

$$\hat{\mathcal{Y}} = \mathcal{C} \times_1 R_U \times_2 R_I \times_3 R_T \tag{2}$$

The n-mode (matrix) product of a tensor $\mathcal{C} \in \mathbb{R}^{j \times k \times l}$ with a matrix $R \in \mathbb{R}^{D \times j}$ multiplies each n-mode tensor fiber by matrix R which denoted by $\mathcal{C} \times_n R$. It is equivalent to multiplying R by the appropriate transformation of tensor $\mathcal{C}$ into matrix C [7]:

$$\widehat{\mathcal{Y}_n} = \mathcal{C} \times_n R \iff \hat{Y}_{(n)} = RC_{(n)} \tag{3}$$

Implementing the general n-mode matrix product for reconstructing tensor on a large dataset is expensive due to memory overflow. The problem becomes worse in the last step of multiplication where the effects of earlier decomposition factors have been included. We propose a memory-efficient loop approach to solve the problem of last iteration. We implement 1-mode and 2-mode (matrix) products to multiply the core tensor $\mathcal{C}$ with the reduced factor matrix R_U and R_I to obtain the intermediate tensor result $\widehat{\mathcal{Y}_1}$ and $\widehat{\mathcal{Y}_2}$ sequentially. To multiply $\widehat{\mathcal{Y}_2}$ with the reduced factor matrix R_T, we implement a 3-mode block-striped (matrix) product. The multiplication task between matrix $\hat{Y}_2$ (the mode-3 matrix equivalent form of tensor $\widehat{\mathcal{Y}_2}$) and R_T is split into N number of subtask, where $N = |T| \; div \; q$ and q is a user-given block-strip row size ($q \ll |T|$). At each subtask, a matrix W_N, where $|W_N| = q$, is obtained by R_T and multiplied with $\hat{Y}_2$. The complete reconstructed tensor $\hat{\mathcal{Y}}$ is achieved by combining all subtask results. The block-stripping of the matrix R_T and multiplication subtasks allow producing smaller manipulations that can fit in the allowed memory size.

The reconstructed tensor $\hat{\mathcal{Y}}$ identifies new entries that are inferred from the latent relationships hidden among the high-order ternary dimensions. It includes a set of triplets $\hat{a}(u, i, t) \in \hat{\mathrm{A}}$ where $A \subset \hat{A}$. Tensor decomposition has recalculated v_A of each existing entry in $\mathcal{Y}$ as well as identified new entries as continues values $v_{\hat{A}}$ in $\hat{\mathcal{Y}}$ which represent the likeliness of user u to tag item i with tag t. Post $\hat{O}$ are assigned binary values $v_{\hat{O}}$ that represent the existence of user-item combinations. As we are generating item recommendation, new item for each user will be selected as new post $\hat{O} - O$ and new tags recommendation will be ignored. Figure 3 shows the process within tensor reconstruction approach used in our proposed method.

Example: Tensor Reconstruction. We explain the stage 1 process with a toy example. Figure 4(a) presents an initial third-order tensor $\mathcal{Y} \in \mathbb{R}^{3 \times 4 \times 4}$ showing 7 tagging and of 6 posting activities. Applying a decomposition technique with 2 as the reduction size results into three reduced-size factor matrices and one core tensor, $R_U \in \mathbb{R}^{3 \times 2}$, $R_I \in \mathbb{R}^{4 \times 2}$, $R_T \in \mathbb{R}^{4 \times 2}$, and $C \in \mathbb{R}^{2 \times 2 \times 2}$. Figure 4(b) shows the reconstructed tensor $\hat{\mathcal{Y}}$ derived by multiplying all decomposed elements. For ease of illustration, we are only showing the top 22 non-zero entries out of a total of 48 entries in $\hat{\mathcal{Y}}$ which grouped as 10 posts. It can be noted that tensor decomposition has recalculated v_A of each existing entry in $\mathcal{Y}$, and identified new entries, as $v_{\hat{A}}$. Since we are interested in recommending items, the process would identify new posts, finding new item for users (ignoring the new tags for existing <user-item> pairs). As highlighted in Figure 4(b), $\hat{\mathcal{Y}}$ generates four (new) items for two users, <2,1>, <2,3>,<3,1> and <3,2>, and utilize them as candidate items to be ranked on the second stage.

2.4 Stage 2: Probabilistic Ranking

This stage takes the new entries (or posts) generated from the reconstructed model and applies probabilistic ranking to rank them as Top-N list of item recommendations. Existing methods rank the candidate items based on the maximum value of $v_{\hat{A}} - v_A$ within every $\hat{O} - O$ [4, 6, 11, 13]. These approaches fail to consider the items and tags usage histories in the tag assignment A, and calculate the recommendations using the level of user preference for an item which based on a tag only. We propose to utilize Naïve Bayes [15] for generating a probabilistic model based on previously observed items and tags usage for ranking the candidate items. We approach the problem of item recommendation as a classification problem, making Naïve Bayes apt for finding an efficient solution [16].

For each user u, based on new posts $\hat{O} - O$ in $\hat{\mathcal{Y}}$, two sets are created. A set of candidate items, $Z_u = \{i_1, i_2, i_3, \dots, i_r\}$ where $Z_u \subseteq I$ that the user u might be interested in, is generated. A tag preference set, $X_u = \{t_1, t_2, t_3, \dots, t_t\}$ where $X_u \subseteq T$ and $|X_u| \leq |\mathrm{T}|$ that user u has used to tag the candidate items, is generated. The tag preference set is generated based on the maximum values of $v_{\hat{A}} - v_A$ which are sorted in descending order. We use the Bayes' theorem for predicting the class candidate item Z_u that have the highest posterior probability given X_u, $p(Z_u|X_u)$. The posterior probability is utilized to calculate the preference probability of user u to select candidate items Z_u by observing the previous usage activities of tag preferences X_u in A. The conditional probability can be formulated as:

$$p(Z_u|X_u) = \frac{p(Z_u)p(X_u|Z_u)}{p(X_u)} \tag{4}$$

where prior $p(Z_u)$ is the prior distributions of parameter set Z_u before X_u is observed; $p(X_u|Z_u)$ is the probability of observing tag preference set X_u given Z_u; and $p(X_u)$ is the probability of observing X_u. Using the assumption of multinomial event model distribution for the Naïve Bayes classifier, the posterior probability p_{u,i_r} of user u with tag preference X_u for candidate item i_r, an instance of Z_u, is obtained by multiplying the prior probability of i_r, $p(Z_u = i_r)$, with the probability of tag preference t_c, an instance of X_u, given i_r, $p(t_c|Z_u = i_r)$:

$$p_{u,i_r} = p(i_r|X_u) = p(Z_u = i_r) \prod_{c=1}^{|X_u|} p(t_c|Z_u = i_r)^{\left(\left(\sum_{i=1}^{|I|} v_{a(u,i_*,t_C)}\right)+1\right)} \tag{5}$$

where $v_{a(u,i_*,t_C)}$ denotes the binary value of assignment A for user u who has used tag preference t_c to tag any item i. The $p(Z_u = i_r)$ and $p(t_c|Z_u = i_r)$ are calculated as:

$$p(Z_u = i_r) = \frac{\sum_{u=1}^{|U|} v_{O(u_*,i_r)}}{\sum_{i=1}^{|I|} \sum_{u=1}^{|U|} v_{O(u_*,i_*)}} \tag{6}$$

$$p(t_c|Z_u = i_r) = \frac{1+\sum_{u=1}^{|U|} v_{a(u_*,i_r,t_C)}}{|T|+\sum_{u=1}^{|U|} \sum_{t=1}^{|T|} v_{a(u_*,i_r,\mathrm{t}_*)}} \tag{7}$$

where $v_{O(u_*,i_r)}$ and $v_{O(u_*,i_*)}$ denote the value of post O for any user u who has tagged candidate item i_r and any item i, respectively. The $v_{a(u_*,i_r,t_C)}$ and $v_{a(u_*,i_r,t_*)}$ denote the value of tag assignment A where candidate item i_r has been tagged by any user u using tag preference t_c and any tag t, respectively. To avoid zero values of Equation 5 and 7, we apply the Laplacean estimate [16] as a smoothing method by adding one to those equations.

For the target user u, the list of Top-N item recommendation is an ordered set of N items, $TopN_u$, obtained by sorting the p_{u,i_r} of user's candidate items in descending order. Figure 5 describes the probabilistic ranking algorithm for generating the Top-N list of item recommendation.

Example: Probabilistic Ranking. The reconstructed tensor as shown in Figure 4(b) generates four new items that correspond to u_2 and u_3. The set of candidate items and tag preferences of u_2 and u_3 are derived as $Z_{u_2} = \{i_1, i_3\}$, $X_{u_2} = \{t_1, t_2, t_3\}$ and $Z_{u_3} = \{i_1, i_2\}$, $X_{u_3} = \{t_4\}$, respectively. Using Equation 5, we calculate the posterior probabilities of u_2 to i_1 and i_3, and of u_3 to i_1 and i_2. Since p_{u_2,i_1}:0.0009 > p_{u_2,i_3}:0.0005, i_1 is more likely to interest u_2 than i_3. While p_{u_3,i_1}:0.0040 = p_{u_3,i_3}:0.0040, i_1 and i_3 are on the same level of interest for u_3. As a result, $TopN_{u_2}$ and $TopN_{u_3}$ are generated in the sequence order of $\{i_1, i_3\}$ and $\{i_1, i_2\}$, respectively. These results differ from the conventional tensor-based approaches which generate $TopN_{u_2}$ and $TopN_{u_3}$ as the sequence order of $\{i_3, i_1\}$ and $\{i_2, i_1\}$, respectively.

3 Empirical Analysis

Two real-world datasets from Delicious (http://delicious.com/) and LastFM (http://www.last.fm/) websites were used. The proposed method is benchmarked with the conventional tensor-based method ("Max") [4] and the state-of-the-art matrix-based method ("CTS") [5]. We demonstrated the variation of our method and the Max method with three commonly used tensor decomposition techniques (i.e. CP, HOOI, and HOSVD [7]) using the Matlab Tensor Toolbox [17]. The results are presented as TRPR-CP, TRPR-HOOI, TRPR-HOSVD, and Max-CP, Max-HOOI, Max-HOSVD for these variations. Adopting the standard practice of decreasing the data sparsity [4, 6, 11, 18], the datasets are refined by selecting users, items, and tags that have occurred in at least p number of posts using the p-core technique [19]. We implemented choices of p-core, as listed in Table 1, to avoid the non-stable results that tend to occur when only one choice of core size is used for the experiments [20].

Table 1. Dataset Statistic

Dataset	p-core	User	Item	Tag	Tag Assignment	Post
Delicious	15	1,609	719	1,761	32,839	17,077
	20	1,359	424	1,321	23,442	12,282
	25	1,198	282	1,053	17,682	9,402
LastFM	10	867	1,715	1,423	99,211	37,163
	20	601	681	838	61,739	22,407
	25	522	490	714	50,381	18,029

We divided the dataset randomly into a training set D_{train} (80%) and a test set D_{test} (20%) based on the number of posts. D_{train} and D_{test} do not overlap in posts, i.e., there exist no triplets for a user-item combination (u, i) in the training set if a triplet (u, i, t_*) is present in the test set. The Top-N items are predicted and ranked for the users present in D_{test}. The performance is measured using F1-Score, as the harmonic mean of overall precision and recall, and reported over the average values as the experiments were implemented using 5-fold cross-validation.

$$Precision(D_{test}, N) = avg_{(u,i)\in D_{test}} \frac{|Test_u \cap TopN_u|}{|TopN_u|} \tag{8}$$

$$Recall(D_{test}, N) = avg_{(u,i)\in D_{test}} \frac{|Test_u \cap TopN_u|}{|Test_u|} \tag{9}$$

$$F1(D_{test}, N) = \frac{2 \cdot Precision(D_{test},N) \cdot Recall(D_{test},N)}{Precision(D_{test},N) + Recall(D_{test},N)} \tag{10}$$

Where $Test_u$ is the set of items tagged by target user in the D_{test} and $TopN_u$ is the Top-N list of items recommended to user from the reconstructed tensor $\hat{\mathcal{Y}}$ which do not exist in the initial tensor $\mathcal{Y}$.

Accuracy: Using F1-score values, we compare the Top-N lists recommendation accuracy between the proposed method and the benchmarking methods. Figure 6 demonstrates that the proposed method outperforms the matrix-based method CTS and the conventional tensor-based method Max. Table 2 lists the average of TRPR recommendation accuracy improvement over the Max method to show the outperformance when implemented on different decomposition techniques used in this paper. The percentage scores are calculated by defining the F1-Scores of TRPR and Max as the current and the starting values, respectively. The scores are reported as an average improvement over all Top-N values. These results ascertain our claim that probabilistically ranking the candidate items, generated from the reconstructed tensor, with utilizing the user's past tagging activities can significantly improve the recommendation accuracy. These results also show the robustness of the proposed method with several decomposition methods. The method with HOOI and CP decomposition techniques achieve bigger improvement than the method with HOSVD. HOSVD optimizes each mode of tensor $\mathcal{Y}$ dimension separately and disregards the interaction among them [7]. Therefore the list of candidate items and tag preferences generated from the reconstructed tensor $\hat{\mathcal{Y}}$ could not reveal the user interest as much as it does for HOOI and CP which use the optimization approach to approximate tensor $\mathcal{Y}$ by taking all lateral interactions into consideration.

It is to be noted that, in general, F1-Scores achieved on offline experiments are low. Our experimental setting may be the reason behind this as the dataset were randomly divided into D_{train} and D_{test} based on the number of posts data. This does not guarantee that for each user in D_{train}, at least one of its post will be selected as D_{test}. Consequently, a target user in D_{test} may not possibly exist in D_{train} (the target user is actually a completely new user).

Table 2. Average TRPR Recommendation Accuracy Improvement over Max Method [4]

Delicious Dataset				LastFM Dataset			
Technique / p-core	CP	HOOI	HOSVD	Technique / p-core	CP	HOOI	HOSVD
15	58.70%	58.24%	11.31%	**10**	28.67%	26.49%	0.25%
20	21.70%	22.31%	4.43%	**20**	26.84%	25.43%	16.90%
25	20.35%	19.56%	1.15%	**25**	32.03%	29.76%	19.38%

Table 3. The Density of the Initial Tensor $\mathcal{Y}$ and Reconstructed Tensor $\hat{\mathcal{Y}}$

Dataset	p-core	Density $(A/(U \times I \times T))$			
		$\mathcal{Y}$ constructed from D_{train}	$\hat{\mathcal{Y}}$ where $\hat{O} - O$		
			CP	HOOI	HOSVD
Delicious	15	0.0014%	0.5239%	0.5110%	0.2753%
	20	0.0027%	1.1113%	1.1178%	0.6594%
	25	0.0043%	1.7970%	1.7916%	1.2622%
LastFM	10	0.0043%	2.1874%	2.2601%	3.2788%
	20	0.0163%	6.7643%	6.9043%	10.6287%
	25	0.0250%	9.4329%	9.3915%	14.3895%

Sensitivity: We examine the effect of choices of p-core to the recommendation accuracy improvement and to the tensor model density. As recorded in Table 2, the p-core size impacts recommendation accuracy. On the Delicious dataset, for all decomposition techniques, the bigger the size of p-core is, the less improvement is achieved. On the contrary, when the LastFM dataset is refined using bigger p-core, the accuracy improvement tend to increase. The characteristic of the datasets is the reason behind this. From Table 1, we can see that, the number of users is always greater than the number of items available on the Delicious dataset. The gap becomes larger as higher p-core is implemented. While on the LastFM dataset, there are sufficient number of items offered for the users resulted from various p-core. Table 3 displays the density of initial tensor $\mathcal{Y}$, built from D_{train}, and reconstructed tensor $\hat{\mathcal{Y}}$. We can examine that the density of $\hat{\mathcal{Y}}$, for all decomposition techniques, is much larger than the density of original $\mathcal{Y}$. Note that the posts in $\hat{\mathcal{Y}}$ which occurred in $\mathcal{Y}$ are excluded, $\hat{O} - O$, since we want to recommend items which have not been tagged by target users. As the purpose of implementing p-core technique is to decrease the dataset sparsity, the trends show that, for all datasets, the bigger the p-core is, the more dense the model.

Scalability: We use space consumption and CPU runtime as the performance metrics to evaluate the scalability of TRPR compared to the Max method [4]. Figure 7 shows the space and time required for tensor reconstruction process of different tensor dimensionalities by varying p-core on the Delicious dataset. Using 15, 50, 80, and 100 core sizes, we built four tensor models of different $user \times item \times tag$ dimensionalities, $1{,}609 \times 719 \times 1{,}761$; $665 \times 52 \times 422$; $362 \times 13 \times 189$; and $250 \times 7 \times 125$, respectively. Accordingly, the bigger p-core size, the smaller the tensor dimensionality is achieved. Figure 7 demonstrates that, for the largest data ($1{,}609 \times 719 \times 1{,}761$), Max failed to run due to memory overflow. The trends show that TRPR is scalable for large tensor size on any decomposition techniques with nearly constant space consumption and a linear time computation to the tensor dimensionality. Consequently, for the purpose of accuracy benchmarking, we had to implement the n-mode block-striped (matrix) product to Max method for making it applicable for all datasets used.

4 Conclusion and Future Work

In this paper, we proposed an item recommendation method using tensor reconstruction approach combined with probabilistic ranking. The method utilizes a memory efficient loop technique for scalable tensor reconstruction and probabilistic ranking to improve the recommendation accuracy of candidate items generated from the reconstructed tensor. Empirical analysis on real-world datasets, with the variations of p-core and decomposition techniques, shows that the proposed method outperforms the benchmarking methods in terms of accuracy and scalability. We have shown that recommendation accuracy can be improved with probabilistically ranking the candidate items, generated from the reconstructed tensor, with utilizing the user's past tagging activities. We also demonstrated that the implementation of n-mode block-striped (matrix) product makes the tensor reconstruction scalable for large datasets. In the future, we are planning to refine the quality of recommendation by implementing clustering approach to solve tag semantic problem.

Reference

1. Zhang, Z.-K., Zhou, T., Zhang, Y.-C.: Tag-Aware Recommender Systems: A State-of-the-Art Survey. Journal of Computer Science and Technology, 26(5): pp. 767-777. (2011).
2. Mezghani, M., Zayani, C.A., Amous, I., Gargouri, F.: A User Profile Modelling using Social Annotations: A Survey. In: The 21st International Conference Companion on World Wide Web. pp. 969-976. Lyon, France: ACM. (2012).
3. Lü, L., Medo, M., Yeung, C.H., Zhang, Y.-C., Zhang, Z.-K., Zhou, T.: Recommender Systems. Physics Reports, 519(1): pp. 1-49. (2012).
4. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: A Unified Framework for Providing Recommendations in Social Tagging Systems Based on Ternary Semantic Analysis. IEEE Transactions on Knowledge and Data Engineering, 22(2): pp. 179-192. (2010).
5. Kim, H.-N., Ji, A.-T., Ha, I., Jo, G.-S.: Collaborative Filtering based on Collaborative Tagging for Enhancing the Quality of Recommendation. Electronic Commerce Research and Applications, 9(1): pp. 73-83. (2010).
6. Rafailidis, D.,Daras, P.: The TFC Model: Tensor Factorization and Tag Clustering for Item Recommendation in Social Tagging Systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 43(3): pp. 673-688. (2013).
7. Kolda, T.,Bader, B.: Tensor Decompositions and Applications. SIAM Review, 51(3): pp. 455-500. (2009).
8. Leginus, M., Dolog, P., Žemaitis, V.: Improving Tensor Based Recommenders with Clustering, in User Modeling, Adaptation, and Personalization, J. Masthoff, et al., Editors. Springer Berlin Heidelberg. p. 151-163. (2012).
9. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag Recommendations based on Tensor Dimensionality Reduction. In: The 2008 ACM Conference on Recommender Systems. pp. 43-50. Lausanne, Switzerland: ACM. (2008).
10. Rendle, S.,Schmidt-Thieme, L.: Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In: The 3rd ACM International Conference on Web Search and Data Mining. pp. 81-90. New York, USA: ACM. (2010).
11. Nanopoulos, A.: Item Recommendation in Collaborative Tagging Systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 41(4): pp. 760-771. (2011).
12. Peng, J., Zeng, D.D., Zhao, H., Wang, F.-y.: Collaborative Filtering in Social Tagging Systems based on Joint Item-Tag Recommendations. In: The 19th ACM International Conference on Information and Knowledge Management. pp. 809-818. Toronto, Canada: ACM. (2010).
13. Kutty, S., Chen, L., Nayak, R.: A People-to-people Recommendation System using Tensor Space Models. In: The 27th Annual ACM Symposium on Applied Computing. pp. 187-192. Trento, Italy: ACM. (2012).
14. Kolda, T.G.,Sun, J.: Scalable Tensor Decompositions for Multi-aspect Data Mining. In: The 8th IEEE International Conference on Data Mining. pp. 363-372. Pisa, Italy: IEEE. (2008).
15. Baker, L.D.,McCallum, A.K.: Distributional Clustering of Words for Text Classification. In: The 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 96-103. Melbourne, Australia: ACM. (1998).
16. Lops, P., Gemmis, M., Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends, in Recommender Systems Handbook. Springer US. p. 73-105. (2011).
17. Bader, B.W., Kolda, T.G., others: MATLAB Tensor Toolbox Version 2.5. Available online, January 2012. http://www.sandia.gov/~tgkolda/TensorToolbox/.
18. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag Recommendations in Folksonomies. In: The 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. pp. 506-514. Warsaw, Poland: Springer-Verlag. (2007).
19. Batagelj, V.,Zaveršnik, M.: Generalized Cores. arXiv preprint cs/0202039, 2002.
20. Doerfel, S.,Jäschke, R.: An Analysis of Tag-recommender Evaluation Procedures. In: The 7th ACM Conference on Recommender Systems. pp. 343-346. ACM. (2013).

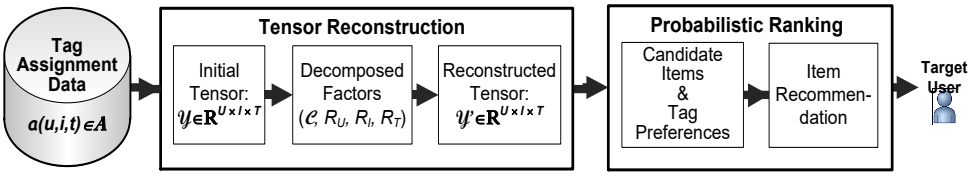


Fig. 1. High-level Definition of the Two-Stage Tensor-based Recommendation

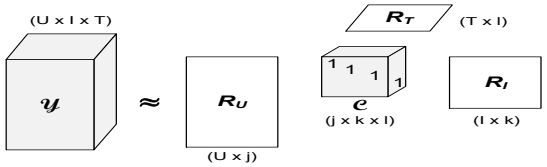


Fig. 2. The CP Decomposition Technique for Third-Order Tensor

Algorithm: ***Tensor Reconstruction***
Input: Tag assignment triplets (A) with $|U|$, $|I|$, and $|T|$ as the number of users, items and tags; Block-strip row size (q) where $q \ll |T|$, $|T|\ div\ q = N$ and $|T|\ mod\ q = b$.
Output: Reconstructed Tensor ($\hat{\mathcal{Y}}$)

1. Construct initial tensor $\mathcal{Y} \in \mathbb{R}^{U \times I \times T}$ from A
2. Apply a decomposition technique to tensor $\mathcal{Y}$ to get:
   a. Left singular factor matrices: M_U, M_I, M_T
   b. Size reduction by choosing: $j \in \{1, |U|\}, k \in \{1, |I|\}, l \in \{1, |T|\}$
      The reduced matrices: $R_U \in \mathbb{R}^{U \times j}, R_I \in \mathbb{R}^{I \times k}, R_T \in \mathbb{R}^{T \times l}$
   c. Core tensor: $\mathcal{C} \leftarrow \mathcal{Y} \times_1 R_U' \times_2 R_I' \times_3 R_T'$ where $\mathcal{C} \in \mathbb{R}^{j \times k \times l}$
3. Reconstruct tensor:
   a. 1-mode (matrix) product: $\widehat{\mathcal{Y}_1} \leftarrow \mathcal{C} \times_1 R_U$, $\widehat{\mathcal{Y}_1} \in \mathbb{R}^{U \times k \times l}$
   b. 2-mode (matrix) product: $\widehat{\mathcal{Y}_2} \leftarrow \widehat{\mathcal{Y}_1} \times_2 R_I$, $\widehat{\mathcal{Y}_2} \in \mathbb{R}^{U \times I \times l}$
   c. **For** $n \leftarrow 1$ to N
      $W_n \leftarrow R_T^{((n-1)q+1,l)}$, $W_n \in \mathbb{R}^{q \times l}$
      3-mode (matrix) product: $\widehat{\mathcal{Y}_{3n}} \leftarrow \widehat{\mathcal{Y}_2} \times_3 W_n$, $\widehat{\mathcal{Y}_{3n}} \in \mathbb{R}^{U \times I \times q}$; $\hat{\mathcal{Y}} \leftarrow \hat{\mathcal{Y}} + \widehat{\mathcal{Y}_{3n}}$
      End for
      If $b \neq 0$ **Then**
      $n \leftarrow n + 1$; $W_n \leftarrow R_T^{((n-1)q+1,l)}$, $W_n \in \mathbb{R}^{s \times l}$
      3-mode (matrix) product: $\widehat{\mathcal{Y}_{3n}} \leftarrow \widehat{\mathcal{Y}_2} \times_3 W_n$, $\widehat{\mathcal{Y}_{3n}} \in \mathbb{R}^{U \times I \times s}$; $\hat{\mathcal{Y}} \leftarrow \hat{\mathcal{Y}} + \widehat{\mathcal{Y}_{3n}}$
      End if /* $\hat{\mathcal{Y}} \in \mathbb{R}^{U \times I \times T}$ */
4. Use entries in $\hat{\mathcal{Y}}$, where $\hat{O} - O$, as candidate items for recommendation

Fig. 3. Tensor Reconstruction Algorithm

U	I	T	v_A	v_O
1	1	2	1	1
1	1	3	1	
1	2	1	1	1
1	3	3	1	1
1	4	4	1	1
2	2	2	1	1
3	4	4	1	1

(a) Initial Tensor $\mathcal{Y}$

U	I	T	v_A	v_O
1	1	1	0.0284	1
1	1	2	1.1169	
1	1	3	1.1189	
1	1	4	0.0001	
1	2	1	0.8796	1
1	2	2	0.0001	
1	2	3	0.0001	
1	2	4	1.0000	
1	3	1	0.9986	1
1	3	2	0.0446	
1	3	3	0.0065	
1	4	4	1.0301	1
2	**1**	**1**	**0.0004**	1
2	**1**	**2**	**0.0085**	
2	**1**	**3**	**0.0085**	
2	2	2	0.9980	1
2	**3**	**1**	**0.0255**	1
2	**3**	**3**	**0.4991**	
2	**3**	**2**	**0.5002**	
3	**1**	**4**	**0.0001**	1
3	**2**	**4**	**1.0000**	1
3	4	4	0.9988	1

(b) Reconstructed Tensor $\hat{\mathcal{Y}}$

Fig. 4. Example of Tensor Model from Toy Dataset with Only Non-zero Values Displayed

Algorithm: ***Probabilistic Ranking***

Input: Initial tensor ($\mathcal{Y}$), Reconstructed tensor ($\hat{\mathcal{Y}}$), Tag preference size (s), Number of Recommendation (N)

Output: The list of N items $TopN_u$

1. Get the candidate item set $Z_u = \{i_1, i_2, i_3, \dots, i_r\}$:
   $Z_u \leftarrow \hat{O} - O$ /*new items in $\hat{O}$ from $\hat{\mathcal{Y}}$ */
2. Get the tag preference set $X_u = \{t_1, t_2, t_3, \dots, t_t\}$ such that ($|X_u| \leq s$):
   $X_u \leftarrow \max_{v_{\hat{A}} - v_A} (\hat{O} - O)$
3. Calculate posterior probability of each item in Z_u and use the value to generate Top-N item recommendation:
   For $r \leftarrow 1$ to N /* initialize the $ListP$ using the first N posterior values of Z_u */
   $p_{u,i_r} \leftarrow p(i_r|X_u)$; $ListP \leftarrow ListP \cup p_{u,i_r}$; $R \leftarrow R \cup r$
   End for
   For $r \leftarrow (N+1)$ to $|Z_u|$
   $p_{u,i_r} \leftarrow p(i_r|X_u)$
   If $p_{u,i_r} > (\min ListP)$ **then**
   $ListP \leftarrow ListP - (\min ListP)$; $R \leftarrow R - r_{min}$
   $ListP \leftarrow ListP \cup p_{u,i_r}$; $R \leftarrow R \cup r$
   End if
   End for
   $TopN_u \leftarrow \{i_r \in Z_u | r \in R\}$

Fig. 5. Probabilistic Ranking Algorithm

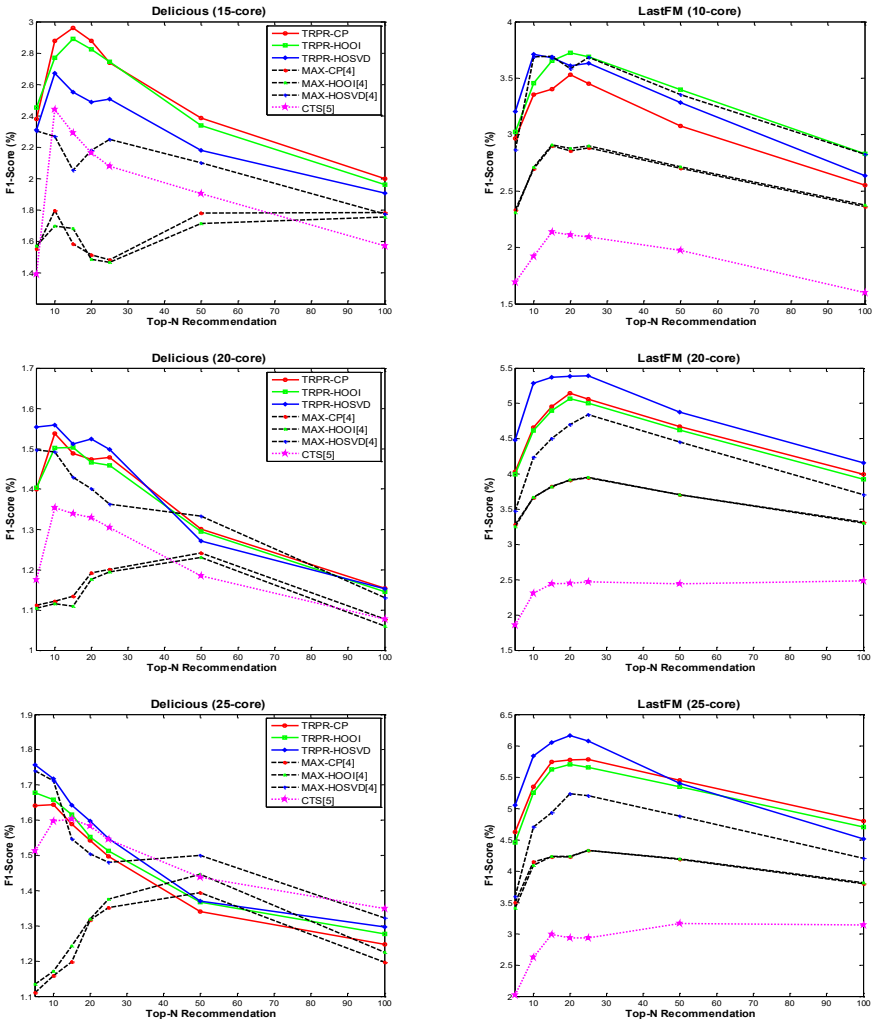
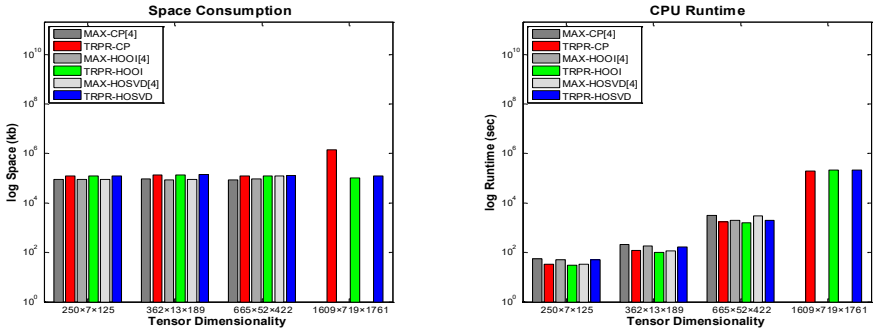


Fig. 6. F1-Score Comparison on Top-N lists for the Recommendation Accuracy



(a) Space Consumption (b) CPU Runtime

Fig. 7. Scalability Comparison by Varying Tensor Dimensionality on Delicious Dataset