D-CARS: A Declarative Context-Aware Recommender System

Rosni Lumbantoruan, Xiangmin Zhou, Yongli Ren, Zhifeng Bao
School of Science, RMIT University, Melbourne, Australia
{rosni.lumbantoruan, xiangmin.zhou, yongli.ren, zhifeng.bao}@rmit.edu.au

***Abstract*—Context-aware recommendation has emerged as perhaps the most popular service over online sites, and has seen applications to domains as diverse as entertainment, e-business, e-health and government services. There has been recent significant progress on the quality and scalability of recommender systems. However, we believe that different target users concern different contexts when they select an online item, which can greatly affect the quality of recommendation, and have not been investigated yet. In this paper, we propose a new type of recommender system, Declarative Context-Aware Recommender System (D-CARS), which enables the personalization of the contexts exploited for each target user by automatically analysing the viewing history of users. First, we propose a novel User-Window Non-negative Matrix Factorization topic model (UW-NMF) that adaptively identifies the significant contexts of users and constructs user profiles in a personalized manner. Then, we design a novel declarative context-aware recommendation algorithm that exploits the user context preference to identify a group of item candidates and its context distribution, based on a Subspace Ensemble Tree Model (SETM), which is constructed in the identified context subspace for item recommendation. Finally, we propose an algorithm that incrementally maintains our SETM model. Extensive experiments are conducted to prove the high effectiveness and efficiency of our D-CARS system.**

***Keywords*-declarative, context-aware, recommender system**

I. Introduction

With the increased development of mobile devices and online services, online activities have become an important part of people's daily life. Users get access to the online sites for shopping, ticket booking and entertainment activities. Huge amount of items are available over online sites, attracting huge number of social users. For example, Amazon has a total of 562,382,292 products as on Jan 10th, 2018, and the number of products is increasing rapidly[1]. In Dec. 2017, Amazon's clothing category has increased by 11%. In 2017, Amazon saw roughly 300 million accounts, and 33 million users who made online purchases using their accounts[2]. Facing such a huge amount of online items, effectively and efficiently navigating the online items becomes impossible for social users. Many companies like Amazon, Google News and Netflix, provided personalized recommendations for this issue. It is believed that contexts greatly affect the recommendation quality. Different contexts are embedded into these systems to enhance their effectiveness. However, we believe that different target users concern different contexts when they select an online item. For example, Anna bought a shirt because of its fashionable design, while Jessie bought the same style of shirt due to the discounted price. Excess contexts will introduce extra time cost, while downgrade the recommendation quality. It is worthwhile investigating more personalized contexts for each target user in recommendation for high system performance.

We study declarative context-aware recommendation for e-business applications. Given a target user u and an item set I, declarative recommendation aims to automatically learn the dominant contexts of u, and return a list of items with the best relevance to u in the dominant context subspace. We focus on effectively and efficiently recommending online items to target users in a more personalized way. This is because the number of online media contexts is very large, and there is the user preference bias with respect to these contexts. For example, an online item may have hundreds of contexts. A user visits a restaurant due to its short distance and atmosphere inside, while others are attracted by its menu price and flavor. However, none of existing solutions has investigated the effect of the context bias on the quality of recommendation. To our best knowledge, this is the first proposal addressing the declarative recommendation problem.

Traditional recommender systems always produce relevant items based on user ratings [1]–[3]. However, problems arise when it goes to items with sparse ratings, while on the other hand the rating sparsity is a common problem on online sites. People made great attempts to improve the recommendation quality. Existing context-aware recommender systems (CARs) use contexts from a representational view, which defines contexts as a predefined attribute set and the structure does not change over time. Though previous CARs have context definitions catered for applications, like the behavioural categories [4], article popularity [5], social trust [6], user interactions and activities [7]–[12], they utilize the pre-specified contexts in data modelling which cannot adaptively capture the dominant contexts for each user, thus cannot guarantee the optimal CARs quality. Motivated by the limitation of current CARs, we proposed a declarative context-aware recommender system (D-CARS) that enables the personalization of contexts for each target user by automatically learning his viewing history over an online site. First, we propose a novel topic model called $user\ window{-}based\ non{-}negative\ matrix\ factorization$ (UW-NMF) to identify the dominant contexts affecting an item purchase of a user by exploiting the reason behind his ratings.

[1]https://www.scrapehero.com/many-products-amazon-sell-january-2018/
[2]https://www.businessinsider.com.au/how-amazons-payments-service-could-solve-its-biggest-weakness-against-paypal-2017-2?r=US&IR=T

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00151

Using these dominant contexts, the profile of this target user is described as a set of transactions in the dominant context subspace. Then, we designed a novel algorithm that learns the predictive model of a target user based on his profile, and conducted a two-phase recommendation by first filtering the item candidates based on the ratings of users having the common dominant contexts with him, and then predicting his preferred items from the candidates using the predictive model. Finally, we propose an incremental update algorithm that maintains the predictive model efficiently. The key contributions of this work are summarized as follows:

1) We propose a new framework that exploits the dominant contexts of a user, the content of the item and the user for declarative recommendation. While the declarative contexts capture the item relevance for a user, ignoring the contexts he does not care avoids extreme time cost.
2) We propose a novel graphical model called UW-NMF to learn the dominant contexts of each target user, based on which the declarative user profile is represented as a set of transactions in the dominant context subspace. The UW-NMF well captures the intent of each transaction.
3) We propose a novel two-phase declarative recommendation that first identifies item candidates and then predicts the final preference using a novel predictive model over a user profile. This avoids the unnecessary operations.
4) We propose a novel algorithm which well maintains the updates in online sites. We conduct extensive tests on two large real datasets to verify the effectiveness and efficiency performance of the proposed solution.

The rest of the paper is organized as follows: Section II reviews the related work on CARs. Section III presents our proposed D-CARs together with its declarative user profiling, two-phase recommendation generation and incremental update maintenance. Experimental results are reported in Section IV. Finally, Section V concludes the whole paper.

II. Related Work

Contexts have been embedded into the collaborative filtering (CF) to enhance the quality of recommender systems [7], [8], [10], [11], [13]–[16]. In [10], Baltrunas et al. exploit time and location of event as contexts and create sub-profiles based on them. Having created two profiles for each user, they used the K-Nearest Neighbour (KNN) to recommend items for each profile based on the ratings in its specific context. In [13], Panniello et al, compared contexts-based pre-filtering and post-filtering and proposed how to pick the best filtering strategy for recommendation. Said et al. apply memory based CF, which create item prediction for a user by grouping similar users and create sub-profiles for each group [11]. Baltrunas et al. proposed a context-aware matrix factorization (CAMF), which introduced three models that represent different assumptions about the interaction of context and ratings [14]. Karatzoglou et al. proposed a Tensor Factorization-based CF by modeling the data as a User-Item-Context N-dimensional tensor [15]. In [16], McAuley et al. justify the user rating by combining latent rating dimensions with latent review topics in their Hidden Factors as Topics (HFT) models. In [8], Yin et al. proposed a temporal context-aware mixture model (TCAM) that models user rating behaviors by considering the user intrinsic interests and temporal context. The TCAM was enhanced by an item-weighting based on the frequency distribution and temporal distribution of items. In [7], Huang et al. proposed a scalable online CF based on the matrix factorization, with an adjustable updating strategy considering implicit user feedback. Though these context-aware CF methods addressed the quality and efficiency issues of recommender systems to different extent, there is still room for the improvement of system performance. All existing approaches select the same set of contexts for all target users when they conduct recommendation, thus the user preference with respect to contexts cannot be captured. As a result, the quality of recommendation cannot be optimized.

III. Declarative Context-Aware Recommendation Framework

We present D-CARS, including declarative user profile construction, recommendation generation, and D-CARS update.

A. Declarative User Profiling

Here, given a user u, his/her declarative profile P_{ui} is defined for each item i s/he rated/reviewed. Specifically, P_{ui} is generated declaratively from u's reviews and the content of both u and item i. The corresponding part from u's reviews is defined as the *user (hidden) context*. Topic models are deployed to reveal these hidden contexts. However, the simple modeling, like Latent Dirichlet Allocation (LDA) [17] is unsuitable for this objective because it does not reveal the hidden topics from each review independently. We propose the UW-NMF (User Window-NMF) based on NMF Topic Modelling, which usually operates on *"documents"* that are defined as the set of all reviews written by user u in this study. Given m users, we have m-disjoint user review-window. Given D_u, the set of reviews from u, and the corresponding set of review terms T_u, we construct $review \times term$ matrix $\mathbf{W}$ for user u, and define w_{ij}, the weight of term t_j in review d_i:

$$w_{ij} = f_{ij} \cdot \left(\log \frac{1 + |D_u|}{1 + |D_u^j|} + 1 \right), \tag{1}$$

where f_{ij} is the frequency of term t_j appearing in review d_i, $|D_u|$ is the number of total reviews from u, D_u^j is the number of reviews containing term t_j. This weighting function is defined in a format of Term Frequency–Inverse Document Frequency (TF-IDF). Then, we applied $L2$ norm to reduce the biased contribution from either common or rare terms:

$$w_{ij} \leftarrow \frac{w_{ij}}{\sqrt{\sum_j^{|T_u|} w_{ij}^2}} \tag{2}$$

Then, NMF was applied in each user window $\mathbf{W}$, which decomposes $\mathbf{W}$ into two k-dimensional latent factors $\mathbf{X}$ and $\mathbf{H}$ with property that all three matrices have no negative elements:

$$\mathbf{X} \times \mathbf{H} \approx \mathbf{W}, \text{ where } w_{ij} \approx \sum_{k=1}^{k} x_{ik} h_{kj}, \tag{3}$$

where $\mathbf{X}$ is an $|D_u| \times k$ matrix, representing the $reviews * topics$ latent subspace; $\mathbf{H}$ is a $k \times |T_u|$ matrix, representing the $topics * term$ latent subspace. We employed Non-Negative Double Singular Value Decomposition (NNDSVD) [18] to enhance the initialization of matrix $\mathbf{W}$ and $\mathbf{H}$. Then, $\mathbf{X}$ and $\mathbf{H}$ are calculated as follows:

$$\mathbf{X} \leftarrow \mathbf{X} \times \mathbf{W}\mathbf{H}^T / \mathbf{X}\mathbf{H}\mathbf{H}^T \quad (a) \quad \mathbf{H} \leftarrow \mathbf{H} \times \mathbf{X}^T\mathbf{W} / \mathbf{X}^T\mathbf{X}\mathbf{H} \quad (b) \quad (4)$$

Note, as $\mathbf{W}$ represents the distribution of terms for each review, $\mathbf{H}$ can be perceived as k-topics defined by the non-negative weights of all terms $t \in T_u$. Then, we can encode

1) to what extent each of the k-topics discussed across all reviews by user u: this is obtained by comparing the rows in $\mathbf{H}$, as each row represents a latent hidden topic.
2) what terms are used to represent the topics: we achieve this by selecting the top-ranked terms based on h_{kj} for each row (topic) in $\mathbf{H}$.

Recalling the motivation of declarative user profiling, we argue that each user has different important contexts when consuming items. Thus, by selecting the top-ranked terms for each topic (row) in $\mathbf{H}$, we avoid setting the same factors for rating and review as well as defining the same contexts for all users. For example, we want to find five topics from user reviews by setting $k = 5$. For each topic, we further choose top 5 ranked-terms that representing each topic. Given the above scenario, we will have at most 25 terms that will be treated as the contexts for the corresponding user.

More importantly, it is infeasible to expect user's preference in all the selected topics in each reviewed/rated item, which means $\mathbf{W}$ is sparse. Then, we define a truncated $\tilde{\mathbf{W}}$ by keeping those selected terms only, termed as *user contexts* for u. Specifically, for each review term $t \in T_u$, we inspect if t appears in the generated the user contexts. If the term t exists in the user contexts, we get term t's value from matrix $\mathbf{W}$ by summing all its value across all reviews as follows:

$$\tilde{w}_{ij} = \sum_{j=1}^{|D_u|} w_{ij} \tag{5}$$

The assumption behind this is that once a user incorporate a context in his review, s/he is considering the similar importance of this context across all his/her reviews. Moreover, we extracted user's and item's contents from metadata, denoted as C_u and C_i. Finally, having user features, we define the declarative user profile for user u on item i as follows:

$$p_{ui} = [\underbrace{\tilde{w}_{u1}, \cdots, \tilde{w}_{ul}}_{\text{user's context}}, \underbrace{c_1^U, \cdots, c_q^U}_{\text{user's content}}, \underbrace{c_1^I, \cdots, c_j^I}_{\text{item's content}}] \tag{6}$$

Note, given u, $\tilde{w}_{u1}$ is the weight of the first ranked term $t \in T_u$ for topic-1 that was *declared* by user u via his/her reviews, which is defined in (5); While c_1^U is the first user u's content, and C_1^I is the first content of item i reviewed by user u. The output of this UW-NMF is a user-window topic model that covers user's contextual preferences. Combining all together with user's and item's contents, we will have declarative user profile. The process details are in Fig. 1.

input: D_u, T_u, k: topic number; **output:** user profile (P)
1. for each user u
2. Initialise $\mathbf{W}$ with (1)-(2), $\mathbf{X}$, $\mathbf{H}$ with NNDSVD [18]
3. update $\mathbf{X}$ and $\mathbf{H}$ with Equ: (4)(a)-(b).
4. select *user context* from $\mathbf{H}$ by selecting the top k ranked terms for each topic (row).
5. for each user u
6. for each review $d \in D_u$
7. for each term $t \in T_u$
8. if t in user contexts
9. calculate $\tilde{w}$ with (5)
10. get the reviewed item's content $[c_1^I, \cdots, c_j^I]$
11. get u's content $[c_1^U, \cdots, c_q^U]$ from metadata.
12. $p_{ui} = [\tilde{w}_{u1}, \cdots, \tilde{w}_{ul}, c_1^U, \cdots, c_q^U, c_1^I, \cdots, c_j^I]$
13. Return $P = \{p_{ui} | u \in U, i \in I, r_{ui} \neq \oslash, d_{ui} \neq \oslash\}$

Fig. 1: Declarative user profiling with UW-NMF

B. Generating declarative Context-Aware Recommendation

In Section III-A, we have constructed the personalized declarative user profiles, which cover user contexts, user content, and the corresponding rated/reviewed item content. In this section, we aim to predict users' preferences towards items given the declarative user profiles. Note that declarative user profile is represented by a set of contexts/contents that is multidimensional arrays. It can be incorporated to the item recommendation generation via two typical techniques: Tensor Factorization (TF) and tree model.

Both TF and tree model produce a predictive model by revealing patterns from the data. However, TF discovers the patterns by identifying the principle low-rank factors of a tensor constructed over the complex content and contexts of users and those of items, which is an extremely expensive process. Each tensor is composed of multiple dimensions, each with a large cardinality. For example, Karatzoglou et. al [15] tried to factorize the tensor over the user, item, and all categorical context variables, which have five variables including seven values in total. The complexity of tensor factorization is exponential to the number of context variables and polynomial to the size of the factorization [19]. For our declarative recommendation, we have 50 contexts, 3 content values for user profiles and 77 content values for items in our datasets. Given a dataset consisting of big numbers of users and items, the dimensionality of resulting tensor over it will be extremely high. This makes the computational cost of the tensor factorization extremely expensive and infeasible for our declarative context-aware recommendation. Moreover, the numerical theory behind TF is ill-posed, which may potentially lead to numerical instabilities [20]. On the other side, tree model has many characteristics that are suitable for our declarative recommendation. It is capable of handling sparse high-dimensional data, and fully utilizes the processor, memory, and disk space to achieve scalable learning. Its success to demonstrate the capability to give the state of the art results was also witnessed in KDDCup 2015, where tree model was used by every winning team in the top-10 [21]. Considering the superiorities of tree model, we select it to produce the predictive model given a declarative user profile. This section proposes a novel algorithm by recommending

items that match with the user's declarative profile. Given user u, D-CARS includes the following components.

Building a SETM (Subspace Ensemble Tree Model): This step aims to learn the user u's consumption pattern based on his/her declarative profiles on all items he/she rated/reviewed, together with the ratings he/she gave to the corresponding items. Specifically, we defined this as a regression problem by treating p_{ui} as the independent variable and the corresponding r_{ui} as the dependent variable. Here, we deployed XGBoost that is based on the tree structure to construct the model. Specifically, considering that each decision in XGBoost is guarded by a feature (each element in p_{ui}), the prediction in tree expansion can be defined as the average value for the training set (the declarative profiles for user u on all reviewed/rated items, P_u) plus the average contribution of each feature [22]:

$$F(p_{ui}) = \frac{1}{J}\sum_{j=1}^{J} f_j(P_u) + \sum_{l=1}^{L}(\frac{1}{J}\sum_{j=1}^{J} contrib_j(p_{ui}, l)) \quad (7)$$

where J is the number of trees and $f_j(P_u)$ is the prediction from the j^{th}-tree with all training data for u; L is the number of features and $contrib_j(p_{ui}, l)$ is the contribution of the l^{th} feature in the feature vector p_{ui} for tree j. The model tries to minimize the objective, $obj(\theta)$, by minimizing the prediction error $\sum_i (r_i - \hat{r}_i)^2$ where θ is the parameters in the trained tree models.

The output is the consumption pattern or the predictive model of the user u: given item $i \in I$, the SETM model is capable of predicting the rating that u is likely to give i. However, considering $|I|$ is normally large, it is not feasible and efficient to make prediction for each item $i \in I$. We proposed to select the subset of items that will be treated as the item candidates by selecting a number of similar users and evaluate their rated items thoroughly.

Selecting the context-aware similar neighbors: Grouping users into a group can be done in many different ways. In this research, we assume that contexts were attached each time a user consume an item, can are captured via the *user context* component in p_{ui}: $[\tilde{w}_{u1}, \cdots, \tilde{w}_{ul}]$. Then, we group the users based on the similarity of their contexts. Given an active user u_a. First we retrieved the declarative user profiling for all users (refer to (6)). Then, we identified u_{group}, a group of users that similar to u_a in accordance with their contexts. In this paper, we deploy k-means to find the similar user groups.

Determining the candidate items to recommend: We retrieved all u_{group}'s previous rated items and define these items, which have not been rated by user u_a, as candidate items. To predict the ratings u_a is likely to give on an candidate item i, we incorporate each candidate item's content to make a temporal declarative profile $\ddot{p}_{ui}$ for u_a:

$$\ddot{p}_{ui} = [\underbrace{\tilde{w}_{u1}, \cdots, \tilde{w}_{ul}}_{\text{user's context}}, \underbrace{c_1^U, \cdots, c_q^U}_{\text{user's content}}, \underbrace{c_1^I, \cdots, c_j^I}_{\text{candidate item's content}} \quad] \quad (8)$$

which includes the user's context, the user's content, and the candidate item's content. By doing so, we put all the information that might influence the preference of users to the items. So, the each of the item candidate, will have the same profile, p_{ui} as in (6), except that the rating information for the candidate item is still missing. However, it will be predicted by using the trained SETM model.

input: user profiling; **output:** recommendation list
1. for each user u
2. Learn SETM model of u_a using the user profile as the training data with the prediction error minimized
3. Retrieve u_a's contexts as in (6)
4. Find u_{group} using k-means clustering. Find candidate items, $I(u_{group})$, the rated items by users in u_{group}.
5. for each candidate item i
6. construct the temporal declarative profile $\ddot{p}_{ui}$ (8)
7. predict $\bar{r}_{uri}$ by using SETM model with $\ddot{p}_{ui}$
8. Rank candidates by the predicted ratings in step 7.
9. Return recommended top-ranked items to user u_a.

Fig. 2: D-CARS item recommendation

Generating recommendations: Then, we proceed to the item prediction by treating each candidate item i's temporal user profile $\ddot{p}_{ui}$ as input to the trained SETM model, and the model outputs the predicted rating $\hat{r}_{ui}$ for u on i. Finally, the D-CARS system will recommend the top N ranked items in the candidate item set to user u.

Overall, the D-CARS item recommendation algorithm can be seen in Fig. 2. In addition, we categorized the deployed model in this phase as an off-line model since we do not update the model with the arrival of new data.

C. Incremental Update

D-CARS works based on declarative user's profiling. Thus, it is a must to retrain the model when new data arrived, unless the performance will degrade. However, retraining the model will be costly in term of time. To solve this problem, we deployed an incremental update of the D-CARS model to adjust with the new arrival data.

We used this data to update the off-line D-CARS model. Specifically, given an off-line model $F(p_{ui})$ (as in III-B) and the arrival of new data D_i. We build a new model $F(D_i)$ that was trained using D_i. Precisely, when training the new SETM model, we use the residual error of the previous tree to improve the performance of the current tree. The intuition behind this incremental update is that the new arrival data D_i will bring the new perspective to the current model. For example, the changes of the dominant features of a user. To cover this needs, we only need to recalculate the importance of each features in terms of coverage and gain. The new weight in the SETM model $\hat{F}(p_{ui})$ was obtained by averaging the old and the new coverage and gain for features in off-line model.

$$\hat{F}(p_{ui}) \leftarrow {}^{(contrib(p_{ui},l)+contrib(\hat{p}_{ui},l))}/_2 \quad (9)$$

where $contrib(\cdot)$ is defined in (7), and $\hat{p}_{ui}$ is the new user profile with the new arriveal data D_i based on (6). Incremental update of D-CARS can be seen in Fig. 3.

input:off-line model, $F(p_{ui})$, $contrib(p_{ui}, l)$
output: updated $\hat{F}(p_{ui})$
1. for each user model u
2. if data D_i exists
3. Construct $F(\hat{p}_{ui})$ over D_i. Get $contrib_j(\hat{p}_{ui}, l))$
4. Find $contrib(p_{ui}, l)$ and $contrib(\hat{p}_{ui}, l)$ average
5. Update $\hat{F}(p_{ui})$ with the new average value in step 4
6. Return $\hat{F}(p_{ui})$

Fig. 3: D-CARS incremental update

TABLE I: Dataset

Dataset	Users	Items	Trans.	AvgReview	Size
Yelp	495	16,618	136K	153	2.23GB
TripAdvisor	2,203	1,640	76K	3	220MG

IV. Experimental Evaluation

A. Experimental Setup

We conduct experiments over 2.45GB data from two real datasets, Yelp and TripAdvisor. Yelp is a subset of the benchmark set from Yelp Challenge Dataset round 10 [23], which includes restaurant review by users. TripAdvisor is a hotel review dataset crawled from www.tripadvisor.com [24]. Their details are described in Table I. Both datasets were created by keeping the users who have rated more than the average reviews. We select 30% of users with the most number of reviews from Table I as the target users. Accordingly, we have 149 target users for Yelp and 661 ones for TripAdvisor. We select the ground truth for each user. The ground truth for a target user are those rated by this user in the test set. We evaluate if the recommendation is successful by calculating the MSE of the predicted rating with the rating in ground truth.

B. Evaluation Methodology

We evaluate the effectiveness and efficiency of our D-CARS. First, we test the effect of parameters to the system effectiveness. Following the optimal topic number setting in [25], we set k to 10. We only evaluate the effect of the number of top terms n. Then, we evaluate the effectiveness of D-CARS using the optimal k and n. For each dataset, we use eight methods to recommend items to the same set of target users, and compare our proposed recommendation, D-CARS10, with the existing five competitors, CTT, SVD++, timeSVD, itemKNN and NMF, and our alternative DCONRS. Finally, we evaluate the effect of incremental model update to the system performance. The approaches for comparison are as follows. (1) DCONRS is our alternative, which generates a user model based on user's preferences towards the contents or property of the items. (2) D-CARS is our proposed method. (3) Incremental D-CARS is our dynamic update model. (4) CTT [7] uses Matrix Factorization to predict ratings for items. (5) SVD++ [1] exploits both ratings and other implicit feedbacks. (6) timeSVD [2] tracks the time changing behaviour of customer throughout the life span of data. (7) ItemKNN [26] recommends items based on the similarity of item consumption of users. (8) NMF [3] only exploits ratings.

The effectiveness is evaluated by a commonly used metric $Mean\ Square\ Error\ (MSE)$ [16], [24]. MSE measures the difference between the set of predicted ratings, $\hat{y}$, and the corresponding ground truth that is the set of ratings, y, in the n-size test dataset. We evaluate the efficiency in term of the time cost of the model maintenance over the whole dataset. All tests are conducted on a server using an Intel Xeon E5 CPU with 256 GB RAM running RHEL v6.3 Linux.

TABLE II: Average MSE vs. n for D-CARS over Yelp

User	n=5	n=10	n=15	n=20
10%	0.644	0.662	0.684	0.734
20%	0.678	0.671	0.690	0.751
30%	0.773	0.765	0.742	0.771
40%	0.808	0.806	0.818	0.857
50%	0.830	0.833	0.858	0.862

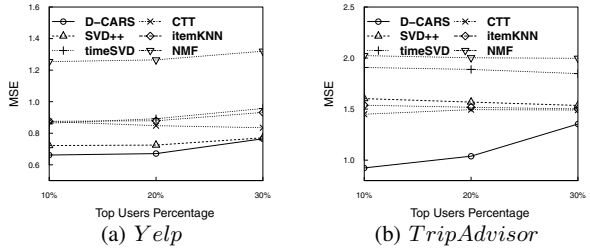
TABLE III: Average MSE vs. n for D-CARS over TripAdvisor

User	n=5	n=10	n=15	n=20
10%	0.925	0.924	0.911	0.921
20%	1.217	1.039	1.039	1.017
30%	1.310	1.354	1.429	1.529
40%	1.705	1.705	1.683	1.707
50%	1.598	1.561	1.574	1.644

C. Experimental Results

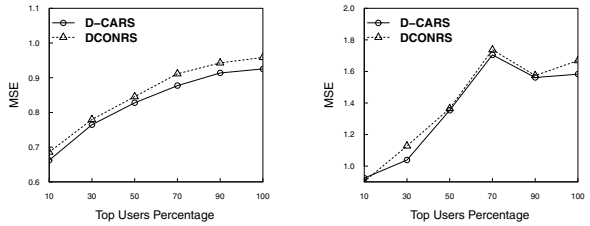
1) Effect of top term number n: We test the effect of n in each topic to the effectiveness of our D-CARS over Yelp and TripAdvisor. In this test, we set k as 10 following the optimal parameter setting suggested in [25]. We decide the optimal n value by varying it from 5 to 20. The average MSE values over two datasets are reported in table II-III. Clearly, there is a no big difference in terms of effectiveness for different n values. However, when n is set to 10, the MSE with respect to top 10%-50% active users performs better than or comparable to other settings. Thus, the default value of n is set to 10.

2) Comparing with existing competitors: We compare the effectiveness of six methods, D-CARS, SVD++, timeSVD, CTT, itemKNN and NMF, by performing recommendation over two datasets to the same sets of target users. We set all methods to their optimal parameter settings, and recommend the predicted items to top 10%-30% target users. We do not investigate other less active users as target ones, since our system aims to solve the problems of recommendation where



(a) $Yelp$ (b) $TripAdvisor$

Fig. 4: Effectiveness comparison with existing competitors



(a) $Yelp$ (b) $TripAdvisor$

Fig. 5: Effectiveness comparison of different alternatives

TABLE IV: Incremental update vs. retrained D-CARS

D-CARS Method	Time	Average MSE
Incremental Update	520.1571 ms	0.8039
Retrained	582.4422 ms	0.7876

the contexts are available. Figs 4a-4b show the comparison of six approaches over Yelp and TripAdvisor respectively in terms of MSE metric. Clearly, our approach achieves the lowest MSE results under all top target user settings for different active target user groups.

Our D-CARS performs better than other competitors due to two reasons. For one thing, our D-CARS captures the more personalized contexts for each target user, which reflect the intension behind each of his transactions, and better capture the user's preference in recommendation. For another, it removes the unimportant contexts in the recommendation to a user, which is a process over a lower context subspace, thus the side effect of redundant contexts can be avoided.

3) Comparing with our alternative: We test the effectiveness of two proposed alternatives: D-CARS and DCONRS. Their comparison results are reported in Figs 5a-5b. Obviously, D-CARS outperformed DCONRS for all users over two datasets. Due to the introduction of personalized dominant contexts, more information can be captured for the preference prediction using our D-CARS. Thus the discrimination power of our D-CARS is higher, leading to a higher effectiveness.

4) Effect of incremental model update: We compare our incremental update model with the retrained model. As shown in Fig. 6, with the incremental update model, the effectiveness of D-CARS is well kept by incrementally updating the model to the new time-slot data. Meanwhile, retrained model achieves the steadily better effectiveness by retraining new data.

We compared the time cost of model maintenance with two methods. The summary of average of MSE and time complexity for model update methods are reported in Table IV. As we can see, our proposed incremental model update method achieves much high efficiency. Considering the effectiveness and efficiency, we can have a better tradeoff of the system.

V. Conclusions

In this paper, we propose a D-CARS that can effectively recommend items over online sites, and can be well maintained efficiently. First, we propose a declarative use profiling that extracts personalized contexts for each of the users based on their review texts. Then we propose a two-phase recommendation algorithm that effectively and efficiently selects the relevant items in the dominant context subspace with respect to a target user. Finally, we propose an incremental model update to maintain the model in a dynamic environment. Extensive experimental results have proved that our proposed approach outperformed the existing methods in terms of efficacy.

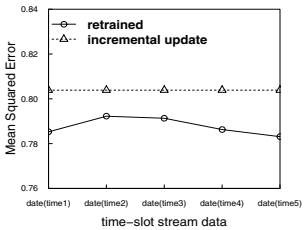


Fig. 6: Effect of incremental model update

VI. Acknowledgment

The research presented in this paper has been supported by the Indonesia Endowment Fund for Education (LPDP).

References

[1] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. SIGKDD*, 2008, pp. 426–434.
[2] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. SIGKDD*, 2009, pp. 447–456.
[3] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *In NIPS*. MIT Press, 2001, pp. 556–562.
[4] M. Jiang, P. Cui, R. Liu, Q. Yang, F. Wang, W. Zhu, and S. Yang, "Social contextual recommendation," in *Proc. CIKM*, 2012, pp. 45–54.
[5] R. Kumar, B. K. Verma, and S. S. Rastogi, "Article: Context-aware social popularity based recommender system," *Int. J. Comput. Appl.*, vol. 92, no. 2, pp. 37–42, April 2014.
[6] X. Yang, H. Steck, and Y. Liu, "Circle-based recommendation in online social networks," in *Proc. SIGKDD*, 2012, pp. 1267–1275.
[7] Y. Huang, B. Cui, J. Jiang, K. Hong, W. Zhang, and Y. Xie, "Real-time video recommendation exploration," in *Proc. SIGMOD*, 2016, pp. 35–46.
[8] H. Yin, B. Cui, L. Chen, Z. Hu, and Z. Huang, "A temporal context-aware model for user behavior modeling in social media systems," in *Proc. SIGMOD*, 2014, pp. 1543–1554.
[9] X. Zhou, L. Chen, Y. Zhang, D. Qin, L. Cao, G. Huang, and C. Wang, "Enhancing online video recommendation using social user interactions," *VLDB J.*, vol. 26, no. 5, pp. 637–656, 2017.
[10] L. Baltrunas and X. Amatriain, "Towards time-dependant recommendation based on implicit feedback," in *(CARS Z09)*, 2009.
[11] A. Said, E. W. D. Luca, and S. Albayrak, "Inferring contextual user profiles – improving recommender performance," 2011.
[12] X. Zhou, L. Chen, Y. Zhang, L. Cao, G. Huang, and C. Wang, "Online video recommendation in sharing community," in *Proc. SIGMOD*, 2015, pp. 1645–1656.
[13] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone, "Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems," in *RecSys*, 2009, pp. 265–268.
[14] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *RecSys*, 2011, pp. 301–304.
[15] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Proc. RecSys*. ACM, 2010, pp. 79–86.
[16] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *RecSys*, 2013, pp. 165–172.
[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[18] C. Boutsidis and E. Gallopoulos, "Svd based initialization: A head start for nonnegative matrix factorization," *Pattern Recogn.*, vol. 41, no. 4, pp. 1350–1362, Apr. 2008.
[19] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proc. SIGIR*, 2011, pp. 635–644.
[20] E. Frolov and I. Oseledets, "Tensor methods and recommender systems," *Wiley Interdiscipl. Rev.: Data Min. Knowl. Discov.*, vol. 7, no. 3.
[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. KDD*. ACM, 2016, pp. 785–794.
[22] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct 2001.
[23] "https://www.yelp.com/dataset_challenge/."
[24] H. Wang, Y. Lu, and C. Zhai, "Latent aspect rating analysis without aspect keyword supervision," in *Proc. SIGKDD*, 2011.
[25] X. Zhou and L. Chen, "Event detection over twitter social media streams," *VLDB J.*, vol. 23, no. 3, pp. 381–400, 2014.
[26] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004.