A Space Alignment Method for Cold-Start TV Show Recommendations

Shiyu Chang*, Jiayu Zhou†, Pirooz Chubak†, Junling Hu†, Thomas S. Huang*
*Beckman Institute, University of Illinois at Urbana-Champaign, IL 61801.
†Samsung Research America, San Jose, CA 95134.
{chang87, t-huang1}@illinois.edu, {jiayu.zhou, p.chuback, junling.hu} @samsung.com

Abstract

In recent years, recommendation algorithms have become one of the most active research areas driven by the enormous industrial demands. Most of the existing recommender systems focus on topics such as movie, music, e-commerce *etc.*, which essentially differ from the TV show recommendations due to the **cold-start** and **temporal dynamics**. Both effectiveness (effectively handling the cold-start TV shows) and efficiency (efficiently updating the model to reflect the temporal data changes) concerns have to be addressed to design real-world TV show recommendation algorithms. In this paper, we introduce a novel hybrid recommendation algorithm incorporating both collaborative user-item relationship as well as item content features. The cold-start TV shows can be correctly recommended to desired users via a so called space alignment technique. On the other hand, an online updating scheme is developed to utilize new user watching behaviors. We present experimental results on a real TV watch behavior data set to demonstrate the significant performance improvement over other state-of-the-art algorithms.

1 Introduction

Recommender systems help to overcome information overload by providing personalized suggestions from a plethora of choices based on the history of user behaviors. While *Collaborative Filtering* (CF) algorithms have seen tremendous achievements in the fields of movie [Koren, 2008; Ning and Karypis, 2011], music [Koenigstein *et al.*, 2011; van den Oord *et al.*, 2013] and e-commerce [Linden *et al.*, 2003] in recent years, limited effort has been conducted to build efficient and effective TV recommender systems [Hu *et al.*, 2008; Xu *et al.*, 2013]. Compared to conventional aforementioned recommender systems, recommending TV shows in practice is more challenging in two aspects:

- **Cold-start**: Since TV shows are live broadcasting, a recommender system needs to predict user preference for upcoming new programs based on watch history. The new TV shows contain no observed user-item relationship, which means all the programs have to be recommended in a cold-start [Zhang *et al.*, 2013; Liu *et al.*, 2013; Ahn, 2008; Elahi *et al.*, 2011; Liu *et al.*, 2011; Xie *et al.*, 2013] manner.

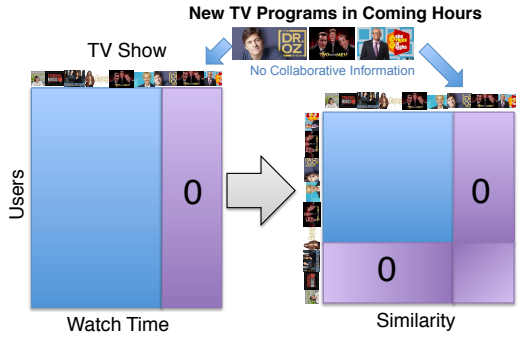


Figure 1: Illustration of the cold-start problem for TV recommendations.

- **Temporal Dynamics**: The user-item matrix is constantly updated as new shows appear over time and new watch data are collected. It is important for a TV recommender system to incorporate new data quickly and update the model online.

An overall scenario for TV recommender systems is illustrated in figure 1, which suggests that the ideal system needs to address above two issues simultaneously. In other words, the desired model should not only be able to recommend unseen items effectively, but also update itself quickly based on the new input data.

Existing methods tackle the problem of cold-start item by involving the auxiliary content information [Forbes and Zhu, 2011; Melville *et al.*, 2002]. These models are known as hybrid models. The hybridization process combines various inputs or composite different recommendation mechanisms. Common approaches combine either *memory-based* algorithms with content similarities [Basu *et al.*, 1998; Melville *et al.*, 2002; Ronen *et al.*, 2013] or encode content information into model-based algorithm learning [Balakrishnan and Chopra, 2012; Forbes and Zhu, 2011; van den Oord *et al.*, 2013].

Due to the unique characteristics of the problem of TV show recommendations, most of the existing memory-based as well as model-based recommendation approaches are still insufficient. The Memory-based methods are efficient but suffer from data sparsity issues. On the other hand, latent-based approaches make use of matrix factorization [Koren,

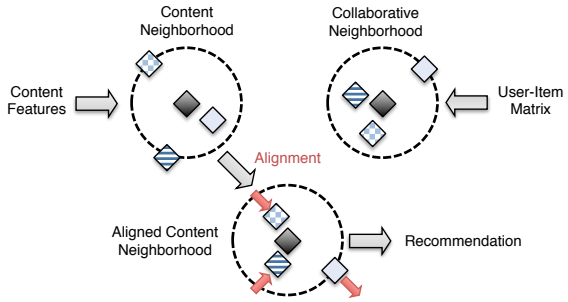


Figure 2: The conceptual illustration of the proposed SAR model.

2008; Chang *et al.*, 2014; Bell and Koren, 2007] to fill the missing data entries by spanning two low-rank matrices. Nevertheless, it requires intensive offline computation, which hardly applies to a real world setting where items dynamically change [Jannach and Friedrich, 2011].

To overcome these difficulties, we develop a novel framework of TV recommender systems termed Space Alignment Recommendation (SAR). SAR is a hybrid model utilizing both CF information via item-item neighborhoods as well as the auxiliary content features for each item. In our tasks, the number of TV shows is sufficiently smaller than the items in other recommendation tasks. The item-item based CF framework provides our model with significant computational advantages over latent factor models in two aspects. First, instead of directly performing model learning on the user-item matrix, our SAR algorithms utilizes the CF knowledge from a much smaller item-item affinity matrix (the number of users is in the range of millions while the number of TV shows are only tens of thousands). Secondly, due to the causality of TV broadcasting, the user-program relationship remains unchanged once the current TV show is over. Item-item similarities calculated from the user-item matrix in the past time stamp are not need to be recalculated.

We refer to the similarity measured from the user-item matrix as the CF space affinity. In contrast, content vectors also provide relative closeness information in a content space. Conventional memory-based hybrid approaches create additional ratings on the cold-start items from the content-based similarity [Melville *et al.*, 2002]. However, the neighborhood relations calculated from the CF space and content space might differ a lot (see the motivating example). Directly incorporating the content knowledge might even hurt the recommendation performance. Unlike these approaches, the basic assumption for the proposed model is that the neighborhood information from the CF space can be approximated by the vector of contents through a transformation. Such a desired mapping can be learned by transferring the knowledge of item-item similarity from the CF space induced by the historical data. Once the embedding function is obtained, the neighborhood information in the content space reflects the corresponding ones in the CF space. Therefore, cold-start TV shows can be recommended systematically. A graphical illustration is shown in figure 2.

Furthermore, in order to handle the rapid data changes, an online updating scheme is presented. This provides SAR the ability to port on real industrial systems with a real-time streaming data. Moreover, it is common for the popular TV shows to have highly correlated neighbors that are based on very few co-watching from viewers. However, these neighbors that are only based on a small number of overlapping users tend to be inaccurate predictors. To devalue such unreliable correlations based on a few user co-watching, our online model assigns each pairwise similarity in CF space a *significance weighting factor* [Melville *et al.*, 2002]. We further discuss the implementation of the proposed algorithm within a MapReduce framework and test it in both batch and online settings. The experimental results show significant advantage of our model compared to other state-of-the-art algorithms in a real large-scale TV show recommendation data set.

2 A Motivating Example

In this section, we describe a toy example from the real cold-start TV recommendation scenario and show why directly adopting content-based similarity as a measure of neighborhood relationship in CF space fails to provide good predictions. We consider the neighborhood measured from two different spaces of the episode of *Dr. Phil* broadcasted on March 13, 2014, which talked about controlling husbands. Table 1 illustrates eight of the most relevant TV shows with *Dr. Phil* in both CF and content spaces. For the CF space, we use the normalized user watch-time to compute the cosine similarity between the TV shows. While for the content space, we extract keywords from the synopsis of the TV shows (for details about content feature extraction, see Evaluation section).

Table 1: Relevant TV shows of *Dr. Phil* measured in both CF and content spaces.

Rank	CF Space	Content Space
1	The Talk	Law & Order
2	Lets Make a Deal	Deadwood
3	The Queen Latifah Show	Love It or List It
4	CBS Evening News	House Hunters
5	The Dr. Oz Show	The Middle
6	The Bold and the Beautiful	My Five Wives
7	The Ellen DeGeneres Show	Fatal Encounters
8	The Price Is Right	Legally Blonde

We observe that among TV shows retrieved from the CF space, most of them are talk shows in various topics. This agrees with the fact about the preference of TV users: people watching the talk show *Dr. Phil* also like other talk shows. In the ranked list from the content space, however, the shows mostly belong to drama. This is mainly because the content of the *Dr. Phil* show in this episode is related to relationship between couples, emotional abuse and victims, and may relate to legal issues. Therefore, the ranking in the content space is biased by such information, and in the ranked list we see shows such as *Law & Order* and *Love It or List It*.

From this example, we see that there is a gap between the CF space and content space: the neighborhoods estimated from the content space cannot reflect the one from CF. As the content space is typically used directly in many existing work for recommending cold start items [Schein *et al.*, 2002; Forbes and Zhu, 2011], it leads to a suboptimal recommendation performance. In this paper, we seek a mapping function

from the content space, such that the gap can be seamlessly bridged and consequently provide a better recommendation performance on cold-start items where only content information is available. We call this procedure *space alignment*.

3 Space Alignment Method for TV Recommendations

Throughout this paper, all matrices are represented by upper case letters (e.g., X and S). The i^{th} row of a matrix X is denoted as X^i, while X_j represents the j^{th} column. We use the capital Greek alphabet to represent sets (e.g., Ω and Λ) and $|\cdot|$ denotes the cardinality of a given set. Moreover, we assume the user-item relationships are given as a utility matrix $X \in \mathbb{R}^{m\times n}$, where m indicates the total number of users and n represents the unique number of items. The observed user-item relationship is given as a set of indices of X represented as Ω. The $(i,j) \notin \Omega$ are the missing entries to be imputed. Furthermore, we denote the temporal factor as an upper index associated with each matrix variable. For instance, M^t represents the matrix variable M at a given time stamp t.

3.1 Space Alignment

We denote the affinity matrix constructed from the CF space as $S \in \mathbb{R}^{n\times n}$, where the (i,j) entry of S can be computed by using pearson correlation or cosine distance. In the content mapping, we want to learn a similarity $s_M(C^i, C^j)$ parameterized in a bi-linear form as $s_M(C^i, C^j) = C^i M (C^j)^T$, which can be achieved by solving the following loss function:

$$\min_M \sum_{i=1}^{n}\sum_{j=1}^{n} \|S_{ij} - s_M(C^i, C^j)\|_2^2 = \min_M \|S - CMC^T\|_F^2. \tag{1}$$

In order to control the complexity of model parameters, we introduce a rank constraint to prevent from overfitting. It provides us the first content alignment algorithm:

$$\min_M \|S - CMC^T\|_F^2, \quad \textit{s.t. } \operatorname{rank}(M) \le r, \tag{2}$$

We note that as a special case, if the M is a positive semidefinite matrix, thus it can be decomposed by $M = LL^T$. In such a case, $L^{d\times r}$ can be seen as a projection from the $d-$dimensional content space to a low-dimensional latent space such that similarities obey the CF space. However, enforcing the positive semi-definiteness is not necessary. As we will show later if the matrix S is positive semidefinite, then the optimal M automatically becomes positive semidefinite.

The optimization problem in (2) can be solved iteratively using projected gradient descent [Nesterov and Nesterov, 2004] or accelerated projected gradient descent. In each iteration, it involves a top-r hard thresholding on the singular values of the matrix variable M. The gradient of equation (2) is given by

$$\nabla_M = -2C^T(S - CMC^T)C = 2C^TCMC^TC - 2C^TSC, \tag{3}$$

where the components C^TSC and C^TC can be computed ahead and stored. In the case of $n \gg d$, computing of gradient is relatively cheap. Now we show that the proposed formulation admits an analytical solution under certain conditions, which is applicable in many recommendation scenarios.

Lemma 3.1 *Assume the content matrix C is in the form of $n \gg d$, and it is full rank. Moreover, the similarity matrix S is symmetric as $S = S^T$. Denote the SVD decomposition of $C = U\Sigma V^T$. Then, the optimal solution of equation (2) is symmetric and in an analytical form as*

$$M^* = VQ\mathcal{H}_r(\Lambda)Q^TV^T \equiv \hat{V}\mathcal{H}_r(\Lambda)\hat{V}^T, \tag{4}$$

where $\mathcal{H}_r(\cdot)$ is the top-r hard singular thresholding [Cai et al.*, 2010]. And $Q\Lambda Q^T$ is the eigen-decomposition of the symmetric matrix $\Sigma^{-1}U^TSU\Sigma^{-1}$.*

Proof Since $n \gg d$ and $rank(C) = r$, there exists a *right inverse* of C denoted as $C^\dagger \in \mathbb{R}^{d\times n}$ such that $C^\dagger C = I_d$ and $C^T(C^\dagger)^T = I_d$. Thus, the objective function in equation (2) is equivalent to

$$\min_M \|C^\dagger S(C^\dagger)^T - M\|_F^2 \quad \textit{s.t. } \operatorname{rank}(M) = r, \tag{5}$$

which admits an analytic solution:

$$M^* = \mathcal{H}_r(C^\dagger S(C^\dagger)^T). \tag{6}$$

We further apply the SVD on the right inverse of C and obtain that $C^\dagger = V\Sigma^{-1}U^T$, where $U \in \mathbb{R}^{n\times d}, V \in \mathbb{R}^{d\times d}$. Therefore we have

$$C^\dagger S(C^\dagger)^T = V(\Sigma^{-1}U^TSU\Sigma^{-1})V^T. \tag{7}$$

It is easy to verify that $\Sigma^{-1}U^TSU\Sigma^{-1}$ is a $d \times d$ real and symmetric matrix, therefore its eigen-decomposition is provided as $Q\Lambda Q^T$. Thus, we conclude the proof.

The core computations to obtain the analytical solution involves an economic SVD and an eigen-decomposition on a $n \times d$ and $d \times d$ matrix respectively, which add up to a $O(d^2(d+n))$ complexity. Moreover, in the case C is not full rank, it is always possible to perform principle component analysis (PCA) to reduce the dimensionality to make the content matrix C full rank. However, PCA imposes a Gaussianized assumption on data distribution, which might lead to a suboptimal solution.

Another approach to handle the problem of rank deficient relies on the *Moore-Penrose pseudo-inverse* instead in the above process. The rank of the optimal solution M^* is jointly determined by $rank(C)$ and r as $\min\{rank(C), r\}$. Therefore, if $rank(C) \le r$, there is no need to perform any truncation and the optimal solution is given by $M^* = C^\dagger S(C^\dagger)^T$. While if $rank(C) > r$, Σ^{-1} is only conducted on the non-zero diagonal entries. Equivalently, $C^\dagger$ can be decomposed as $V'\Sigma'^{-1}U'^T$, where $V' \in \mathbb{R}^{d\times rank(C)}$ and $U \in \mathbb{R}^{n\times rank(C)}$. It is worth mentioning that, in such a case, perform the eigen-decomposition on $\Sigma'^{-1}U'^TSU'\Sigma'^{-1}$ and then apply truncation on the eigen-value gains significant computational advantage over directly using equation (6). The reason is that instead of conducting SVD on a $d \times d$ matrix, the eigen-decomposition allows us to work with $r \times r$ space. On a separate note, from the analytical solution in (4), we can easily verify that M^* is guaranteed to be a PSD matrix if S is PSD.

After the content mapping M^* is obtained, we are able to use it to recommend the cold-start shows. For completeness of the paper we briefly describe this procedure [Linden

et al., 2003] as follows. Let $\mathcal{P}$ be the set containing indices of shows broadcasted in the past, while $\mathcal{F}$ represents the set of cold-start shows in the future time stamp. We generate the item-item similarity between shows in $\mathcal{P}$ and $\mathcal{F}$ as $s_{p,f} = C^p M^* (C^f)^T,\ \forall p \in \mathcal{P},\ f \in \mathcal{F}$. Then, the imputation of the future watching activity is given as

$$X_{if} = \frac{\sum_{p \in \mathcal{T}_{ipf}(N)} X_{ip} \cdot s_{pf}}{\sum_{p \in \mathcal{T}_{ipf}(N)} s_{pf}}, \forall\, i \in \{1, \ldots, m\}, f \in \mathcal{F}, \quad (8)$$

where, $\mathcal{T}_{ipf}(N)$ is the set that indicates the top-N most similar items to the query f among all candidate items in $\Omega_{i\cdot}$ for the particular user i. $\Omega_{i\cdot}$ is denoted as the subset of Ω with user i fixed. Once the missing entries are imputed, we are able to rank all TV shows in $\mathcal{F}$ for each user to perform top-k recommendations.

3.2 Weighted Space Alignment

The model in equation (2) penalizes the difference between the given CF similarity and transformed content similarity at every pair of $(i, j),\ \forall i, j \in \{1, \ldots, n\}$. However, the Forbinius norm treats each pair of inputs equally important. We generalize the equal weight assumption by introducing a *reliability matrix* $R \in \mathbb{R}^{n \times n}$, where each (i, j) entry of R represents the weight emphasis of the reconstruction. The more weight appears in an entry of R, the more penalty is obtained for the specific pair of inputs.

In general, R can be constructed from the prior information by domain experts. However, in the field of recommendation, R is more commonly used to reflect the trustworthiness of affinity computed from the CF space. A similar idea of significance weighting factor [Herlocker *et al.*, 1999] can be adopted. Recall that CF similarity calculation makes use of a pair of columns in the utility matrix. Two popular TV shows may contains hundreds of thousands of co-ratings (supports), while some are only based on a few. We weigh the similarity computed from a large number of supports a high weight, and vise versa. Therefore, the loss function of the proposed method can be modified to reflect the trustworthiness in the following manner.

$$\min_M \sum_{j=1}^{n} \sum_{i=1}^{n} R_{ij} (S_{ij} - C^i M (C^j)^T)^2 = R \circ \|S - CMC^T\|_F^2. \quad (9)$$

where $\circ$ denotes the Hadamard product. and the gradient is given by

$$\begin{aligned} \nabla_M &= -2C^T \Big(R \circ (S - CMC^T) \Big) C \\ &= 2C^T (R \circ CMC^T) C - 2C^T (R \circ S) C. \end{aligned} \quad (10)$$

There is no analytical form due to the Hadamard product. However, the model can be solved using the projected gradient descent [Nesterov and Nesterov, 2004].

3.3 Online Modeling for Temporal Dynamics

In the setting of the TV recommendation, the recommender system constantly receives new watch information from the users, and we would like to update the model to reflect the latest information. This temporal dynamic setting imposes challenges from the perspective of computation, as it is infeasible to re-train a model using all previous data when new inputs are available. To solve this challenging problem, we resort to online approaches. In contrast to offline approaches, which assume all user watching information is provided up front, online models assume that information is received one piece at a time, and only utilize the newly available data to update a previous model.

Specifically, we assume the content similarity function is parameterized as M_t at a given time stamp t. The new-coming information is received as a quaternion $((C^i)^t, (C^j)^t, S_{i,j}^t, R_{i,j}^t)$. Given the model M_t, we would like to update the model parameters based on the quaternion, while on the other hand we want the model to be relatively stable, i.e., the model should not change too much after observing the newly available supervision information. Based on the two criterion, the model proposed in equation (9) can be naturally extended as

$$M_{t+1} = \text{argmin}_M R_{ij} (S_{ij}^t - (C^i)^t M ((C^j)^t)^T)^2 + \alpha \mathcal{D}(M_t, M), \quad (11)$$

where $\mathcal{D}(M_t, M)$ is a regularization function, and α is a control parameter to place an emphasis between the two criterion we mentioned above. Note that in this online formulation we have dropped the rank constraint from the offine model in order to adapt to fast incoming online data. The overfitting can be controlled by the regularization term, which requires the solution M_{t+1}^* to be close to a low-rank solution we obtained in the offline step, or a solution we obtained at the previous time stamp. Readers are referred to [Davis *et al.*, 2007; Jain *et al.*, 2008; Shalev-Shwartz *et al.*, 2004] for details of the online learning.

4 Experiments

4.1 Experiment Settings

The data we report upon in this paper comes from the server logs at Samsung where the data is anonymized and encrypted for privacy reasons. We randomly select a subset of users to prevent confidential disclosure over 15 consecutive days, which contains 1,697,374 active users and 17,553 unique TV shows. It is worth pointing out that the number of items in our settings is much smaller compared to the e-commerce or media recommendations. The observation is that typical users watch less than 20 TV shows each day (93% users) while only very few of users watch more than 40 shows (0.3% users).

To extract informative and discriminative content features, we follows the strategy proposed by [Phan *et al.*, 2008] by utilizing 200 latent topics extracted from Wikipedia where each topic is associated with 200 keywords. Based on these latent topics, we present each TV show to a 200-dimensional vector by using its title, description and genre (after tokenizing and stemming). In addition, we concatenate it with channel, program starting time and ending time as three additional features. The final dimensionality of each TV program in the content space is 203. We further normalize the content vector to make the the ℓ_2-norm to one.

To evaluate the quality of the recommender systems, we split these 15 days of data into eight folds, each contain-

ing eight consecutive days of data. Then we use the first seven days as the training set and the last day as a cold-start testing set. By doing so, our settings will be the same to the real world TV show recommendation scenario as recommending the next day TV shows by using the data of the past week. Furthermore, we report the recommendation performance measured by the top-k mean average precision (MAP) and recall (MAR) [Herlocker *et al.*, 2004] framework and all the reported results are averaged over eight folds.

4.2 MapReduce Implementation

In this section we discuss how the proposed SAR algorithm handles recommendation tasks at large-scale using the MapReduce framework.

Training Model: In training the offline model, the bottleneck of efficient learning is to calculate the pair-wise affinity matrix from the utility matrix. Nevertheless, as the computation is independent for each pair of TV shows, it can be efficiently computed in a parallel fashion. A detailed tutorial on item-item similarity computing within the Hadoop framework can be found in [Apache,]. Once the similarity matrix S is obtained, training the offline model only involves top SVD, which can be readily solved by the Apache Mahout framework [Apache,].

Making Recommendations: Similar to the training stage, the similarity between every pair (i, j) $\forall i \in \mathcal{P}, j \in \mathcal{F}$ can be calculated distributively by caching the model M^* in every reducer. The new generated similarity matrix between past TV shows and cold-start programs can be again easily adapted to the existing top-k recommendation settings within Apache Mahout framework.

4.3 Parameter Study

Choice of Rank Constraint: Before proceeding to evaluate the recommendation performance, we first inspect the quality of neighborhood reconstructions to validate our assumption. Since the goal of the proposed algorithm is to learn a good similarity measure from content features that is close to the one from the CF space, the ideal performance can be achieved when the program neighborhoods (the top-p most similar items) computed from the content features can perfectly coincide with the ones computed from the user watch history. Therefore, the assumption of the proposed SAR method is validated by the percentage of neighborhood coverage from the content space to the CF space. We use the cosine similarity between each pair of content vectors in the original content space as a baseline method to compare with, and the result is shown in figure 3.

We compute the neighborhood coverage on the testing day and vary the rank from five to 50 for the purposed method. We note that SAR significantly improves the neighborhood coverage. By setting the rank as 45 in equation (2), we are able to achieve around 50% neighborhood overlap, while the performance of the baseline is less than 10%.

Another observation is that the performance of neighborhood reconstruction of the proposed methods continuously improves as the rank of M increases until the rank reaches 45 and start to drop after that. This is because the optimal solution of the SAR method starts to overfit the training data and loses the generalization ability. Throughout this paper, we set the rank of M in the range of 40-50.

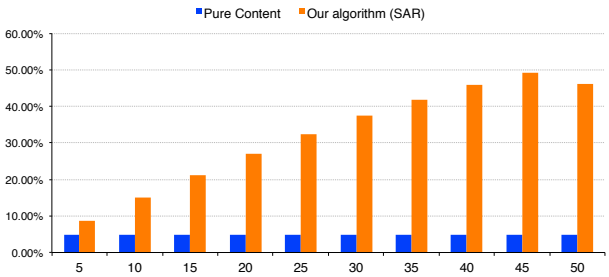


Figure 3: Results on neighborhood reconstruction.

Choice of Neighorhood Size: The proposed SAR method is built upon the item-item CF approach, where the missing value is imputed by a weighted average of a neighborhood given by the similarity matrix. The size of the neighborhood N is a parameter to be determined, which has significant impact on the recommendation performance [Herlocker *et al.*, 1999; Sarwar *et al.*, 2001]. We vary N from 5 to 100 and present the performance in terms of top-k MAP and MAR for different k values. The graph illustration is presented in figure 4. We see that for different k values, the best MAP and MAR are achieved when the neighborhood size is range of 30-50. We use a neighborhood size $N = 40$ through all our experiments.

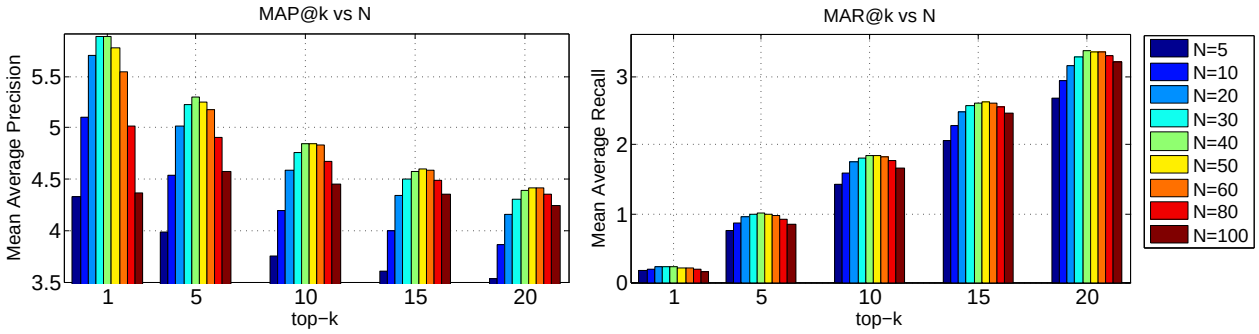
4.4 Recommendation Performance

Related Methods: We compare the proposed algorithm with other cold-start techniques. The detailed descriptions of the other baseline methods are as follows:

- **CMF** [Forbes and Zhu, 2011]: Content-boosted Matrix Factorization handles the cold-start problem by assuming the latent item profile as projection of content features.
- **A-NMF** [Gantner *et al.*, 2010]: Attribute-to-feature mapping based on the Non-negative Matrix Factorization (A-NMF). It performs NMF [Lee and Seung, 2000] on the training data to obtain user and item latent factors. A regression model is trained from the content space to latent item space to leverage cold-start recommendations.
- **A-PMF** [Gantner *et al.*, 2010]: Similar to A-NMF, the only difference is that A-PMF relies on the Probabilistic Matrix Factorization (PMF) model [Salakhutdinov and Mnih, 2007] to obtain the latent factors.

The parameters of all competing methods are determined by a 10% validation set separated from the training data.

Results: We compare the proposed SAR algorithm with other state-of-the-art methods and report the expermiental results in table 2. In the experiment, we vary top-k from 1 to 30 and observe that the proposed SAR method significantly outperforms existing methods in terms of both precision and recall. Among the other baselines, CMF achieves the best performance followed by A-NMF and then A-PMF. One possible reason for A-NMF and A-PMF obtaining the worst recommendation performance is that matrix factorization based approaches predict the user likeness by both user and item la-



(a) Top k mean average precision

(b) Top k mean average recall

Figure 4: The top k mean average precision (MAP@k) and recall (MAR@k) given different size of neighborhood N. At all k values, the best performance is obtained using a neighborhood around size 40.

Table 2: Recommendation performance in terms of top-k mean average precision (P) and recall (R). Both precision and recall values are shown in percentage.

Top@N		@1	@5	@10	@20	@30
A-PMF	P	4.33	3.97	3.75	3.53	3.40
	R	0.16	0.75	1.42	2.68	3.88
A-NMF	P	4.61	4.08	3.85	3.59	3.47
	R	0.17	0.79	1.45	2.73	3.91
CMF	P	4.91	4.34	3.93	3.68	3.54
	R	0.18	0.81	1.51	2.82	4.03
SAR	P	**5.89**	**5.30**	**4.84**	**4.39**	**4.12**
	R	**0.22**	**1.01**	**1.84**	**3.38**	**4.70**

tent factors. However, both A-NMF and A-PMF are two step approaches that rely on a regression model conducted on pre-factorized item factors only. It essentially overlooks the user aspect of the rating prediction, which results in a poor performance.

CMF handles the cold-start problem in an integrated framework. However, comparing to the proposed SAR method at top-1 precision, its performance is almost 1% lower. One reason that the proposed SAR approach can outperform CMF may be due to the model complexity. From the perspective of the degree-of-freedom, the SAR method learns a mapping matrix of size $d \times r$ given an input $n \times n$ similarity matrix. While the CMF estimates a $d \times r$ mapping matrix as well as a user profile matrix of size $m \times r$ from $|\Omega|$ number of inputs, where $|\Omega|$ is the number of non-zero elements in the user-item matrix X. Especially, in our experiment settings, we have $m = 1.69$M, $d = 203$, $n = 17,553$ and $|\Omega| = 83.6$M. Given both SAR and CMF set the rank of matrix as 50, SAR estimates $d \cdot r = 10$K parameters from $n^2 = 308$K observations, while $(d+m)r = 84.8$M parameters have to be learned in the CMF model from $|\Omega| = 83.6$M inputs.

Temporal Dynamics: In this section we study the performance of the online evolving model in the TV show recommendation with temporal dynamics. We compare two recommendation settings: the offline SAR model without evolving, and SAR model with online evolving (SAR Evolve). In the former setting, we select the data from the first seven days to learn an offline content mapping parameter M, and use it to recommend TV programs for the next eight consecutive days. On the other hand, the evolving model makes use of the output from the offline model trained by the first seven days as the initialization point. It updates the parameter M before it proceed to recommend the TV shows for the next day. For example, before the evolving model performs recommendation for day 9, we update the model using the 8th day's data in an online manner. In both settings we use the weighted SAR model, where the weight value is given by the number of supports (shared users) divided by a normalization factor [Melville *et al.*, 2002].

Table 3: The recommendation performance of offline SAR model and online SAR evolving model, in terms of top-1 mean average precision. The online model consistently outperforms the offline model.

Top-1 MAP	day 8	day 9	day 10	day 11
SAR	5.86	5.78	5.80	5.69
SAR Evolve	5.86	5.81	5.81	5.77
Top-1 MAP	day 12	day 13	day 14	day 15
SAR	5.82	5.77	5.65	5.81
SAR Evolve	5.84	5.80	5.72	5.82

We present the experimental results for top-1 precision in Table 3. We see that the online SAR evolving approach delivers consistently better performance than the offline approach. However, we notice that the performances of both models are not stable, i.e., at day 11 and day 14 the performances are relatively lower than those at other days. This might relate to the periodic repeat patterns of certain TV shows.

5 Conclusion

In this paper, we proposed a novel hybrid TV show recommendation algorithm termed SAR , which incorporates both collaborative user-item relationship as well as item content features. SAR seeks a mapping from the content space to the CF space so that the similarity between content features is close to the one computed from the utility matrix. In addition, an online evolving model is proposed, which is not only able to refine the model parameter from the rapid input changes but also considers the trustworthiness of the neighborhood relations. The intensive experimental results provide strong evidence that the SAR method outperforms other state-of-the-algorithms in the field TV recommendations.

References

[Ahn, 2008] Hyung Jun Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[Apache,] Apache. Apache mahout:: Scalable machine-learning and data-mining library.

[Balakrishnan and Chopra, 2012] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *WSDM*, pages 143–152. ACM, 2012.

[Basu *et al.*, 1998] Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI*, pages 714–720, 1998.

[Bell and Koren, 2007] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[Cai *et al.*, 2010] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[Chang *et al.*, 2014] Shiyu Chang, Guo-Jun Qi, Charu C. Aggarwal, Jiayu Zhou, Meng Wang, and Thomas S. Huang. Factorized similarity learning in networks. In *ICDM*, pages 60–69, 2014.

[Davis *et al.*, 2007] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216. ACM, 2007.

[Elahi *et al.*, 2011] Mehdi Elahi, Valdemaras Repsys, and Francesco Ricci. Rating elicitation strategies for collaborative filtering. In *E-Commerce and Web Technologies*, pages 160–171. Springer, 2011.

[Forbes and Zhu, 2011] Peter Forbes and Mu Zhu. Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In *Recsys*, pages 261–264. ACM, 2011.

[Gantner *et al.*, 2010] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, pages 176–185, 2010.

[Herlocker *et al.*, 1999] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237. ACM, 1999.

[Herlocker *et al.*, 2004] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *TOIS*, 22(1):5–53, 2004.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. IEEE, 2008.

[Jain *et al.*, 2008] Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *NIPS*, volume 8, pages 761–768, 2008.

[Jannach and Friedrich, 2011] Dietmar Jannach and Gerhard Friedrich. Tutorial: Recommender systems. In *IJCAI*, 2011.

[Koenigstein *et al.*, 2011] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Recsys*, pages 165–172. ACM, 2011.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD*, pages 426–434. ACM, 2008.

[Lee and Seung, 2000] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[Liu *et al.*, 2011] Nathan N Liu, Xiangrui Meng, Chao Liu, and Qiang Yang. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *Recsys*, pages 37–44, 2011.

[Liu *et al.*, 2013] Haishan Liu, Mohammad Amin, Baoshi Yan, and Anmol Bhasin. Generating supplemental content information using virtual profiles. In *Recsys*. ACM, 2013.

[Melville *et al.*, 2002] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI*, pages 187–192, 2002.

[Nesterov and Nesterov, 2004] Yurii Nesterov and IU E Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

[Ning and Karypis, 2011] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*, pages 497–506. IEEE, 2011.

[Phan *et al.*, 2008] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, pages 91–100. ACM, 2008.

[Ronen *et al.*, 2013] Royi Ronen, Noam Koenigstein, Elad Ziklik, and Nir Nice. Selecting content-based features for collaborative filtering recommenders. In *Recsys*, pages 407–410. ACM, 2013.

[Salakhutdinov and Mnih, 2007] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, volume 1, pages 2–1, 2007.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. ACM, 2001.

[Schein *et al.*, 2002] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260, 2002.

[Shalev-Shwartz *et al.*, 2004] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. Online and batch learning of pseudo-metrics. In *ICML*, page 94. ACM, 2004.

[van den Oord *et al.*, 2013] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *NIPS*, pages 2643–2651, 2013.

[Xie *et al.*, 2013] Yusheng Xie, Zhengzhang Chen, Kunpeng Zhang, Chen Jin, Yu Cheng, A. Agrawal, and A. Choudhary. Elver: Recommending facebook pages in cold start situation without content features. In *IEEE Big Data*, pages 475–479, Oct 2013.

[Xu *et al.*, 2013] Mengxi Xu, Shlomo Berkovsky, Sebastien Ardon, Sipat Triukose, Anirban Mahanti, and Irena Koprinska. Catch-up tv recommendations: show old favourites and find new ones. In *Recsys*, pages 285–294. ACM, 2013.

[Zhang *et al.*, 2013] Xi Zhang, Jian Cheng, Ting Yuan, Biao Niu, and Hanqing Lu. Semi-supervised discriminative preference elicitation for cold-start recommendation. In *CIKM*, pages 1813–1816, 2013.