Towards a Dynamic Top-N Recommendation Framework

Xin Liu
École Polytechnique Fédérale de Lausanne
(EPFL)
Lausanne, Switzerland
x.liu@epfl.ch

Karl Aberer
École Polytechnique Fédérale de Lausanne
(EPFL)
Lausanne, Switzerland
karl.aberer@epfl.ch

ABSTRACT

Real world large-scale recommender systems are always dynamic: new users and items continuously enter the system, and the status of old ones (e.g., users' preference and items' popularity) evolve over time. In order to handle such dynamics, we propose a recommendation framework consisting of an online component and an offline component, where the newly arrived items are processed by the online component such that users are able to get suggestions for fresh information, and the influence of longstanding items is captured by the offline component. Based on individual users' rating behavior, recommendations from the two components are combined to provide top-N recommendation. We formulate recommendation problem as a ranking problem where learning to rank is applied to extend upon matrix factorization to optimize item rankings by minimizing a pairwise loss function. Furthermore, to better model interactions between users and items, Latent Dirichlet Allocation is incorporated to fuse rating information and textual information. Real data based experiments demonstrate that our approach outperforms the state-of-the-art models by at least 61.21% and 50.27% in terms of mean average precision (MAP) and normalized discounted cumulative gain (NDCG) respectively.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

top-N recommendation; learning to rank; matrix factorization; topic modeling

1. INTRODUCTION

Most large-scale recommender systems are highly dynamic, where new users and items continuously enter the system, and existing users' preference and items' popularity may vary over time. In order to handle such dynamics for personalized recommendation, a set of time-aware models have been proposed, which employed decay function to give more weight to recent observations [15], or considered only information in specific time-windows [14], or incorporated temporal effects into latent factor models [13, 25]. However, existing solutions either only focus on local information (e.g., recent data or data in certain time-windows) thus *deemphasizing* the influence of global information (e.g., the longstanding items) which might also reflect users' current preference (e.g., seasonal/periodic preference) [3], or suffer from the scalability issue, e.g., frequently re-training latent factor model over big data is computationally expensive.

In order to address the issues of existing time-aware recommendation models, we propose a flexible framework that takes into account both fresh information and longstanding information but processes them separately. The framework consists of an online component and an offline component, where the online component frequently updates the online recommendation model such that users are able to timely receive recommendations for continuously arrived items, and the offline component slowly updates the offline recommendation model to capture the influence of (typically huge volume) longstanding items[1]. When a recommendation query is issued, two recommendation lists from the two components are generated and combined to provide the final top-N recommendation to the user based on her past rating behavior (i.e., preference on the freshness of items).

For each component (online or offline), in contrast to the state-of-the-art solutions that focus on achieving high rating prediction accuracy, we model the recommendation problem as a ranking problem. Specifically, we apply learning to rank technique on the basis of matrix factorization to optimize recommendation ranking by minimizing a *pairwise* loss function. The rank of a pair of items is based on their relevance, where the relevance is determined by the users' preference on items, characterized by the corresponding user's and items' latent factors.

Furthermore, in addition to rating information, we consider textual contents which are pervasive in recommender systems (e.g., tags, descriptions of items) to more precisely model users' preference on items. In this setting, each item is represented by a "bag-of-words" and the words of all items

[1]In this work, we consider a generic recommendation scenario where every item is potentially a candidate for recommendation like movie and book recommendation. For the scenario where items have short lifespan (e.g., news recommendation), we can remove outdated items periodically *or* just use online component.

RecSys'14, October 06 - 10 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645720.

construct a text corpus. We apply latent dirichlet allocation (LDA) [4] to extract a set of topics from the corpus, and assign each topic a latent factor vector that shares the same space with that of users and items. A user's preference on an item is inferred by combining (1) the user's affinity to topics, derived by multiplying user-specific and topic-specific latent factors and (2) the correlation between topics and the item, i.e., the topic distribution of the item inferred by LDA. This is different from the state-of-the-art approaches [22, 17] that reconcile the number of topics and the dimensionality of latent factors (i.e., implicitly assume that textual content is the dominant factor that influences the distribution of latent factors[2]), thus is able to more flexibly incorporate textual contents into matrix factorization.

The contributions of this paper are summarized as follows: (1) We propose a recommendation framework consisting of an online component and an offline component to process the continuously arrived items and longstanding items separately. (2) We fuse learning to rank and latent factor model to provide top-N recommendation by optimizing a pairwise loss function. Additionally, we apply LDA to better model users' preference for item ranking. To the best of our knowledge, this is the first work that integrates topic modeling into learning to rank model to improve top-N recommendation. Note that such a recommendation model is used by both online an offline components. (3) We cluster each user's rated items based on their topic distribution such that the items in the same cluster can be more meaningfully compared for optimizing pairwise loss function. Moreover, such a clustering strategy removes the item pairs that are not meaningfully comparable, thus reducing the training data while keeping its informativeness. (4) We evaluate the performance of the proposed recommendation method over real datasets. Experimental results show that our approach significantly outperforms the state-of-the-art models.

2. RELATED WORK

Time-aware recommendation

Traditional neighborhood-based algorithms handle temporal effect by either giving more weight to recent information or focusing on observations in specific time windows. For instance, in [25], a decay function $f(x) = e^{-\alpha t}$ was utilized to reduce the effect of old ratings, where t is the time at which a rating was given and α is the decay rate. Liu. et al. [15] used the similar decay function for both similarity computation and rating prediction.

For latent factor models, Koren [13] tackled temporal effect by using time function to model user and item biases in matrix factorization. In [25], the authors split time in equal time intervals and added the time dimension to rating matrix to form a tensor. Alternating least square algorithm was applied to optimize the tensor model.

Recently, a series of solutions were proposed to particularly process the highly dynamic data stream for social media like Twitter. In [7], the authors proposed a collaborative ranking algorithm to recommend interesting tweets to users taking into account tweet topic level factors, user social relation factors, explicit features and the quality of tweets. In [9], tweet topic recommendation was studied, and a ranking algorithm called Stream Ranking Matrix Factorization was proposed where a pairwise ranking approach was applied to optimize the personalized ranking of topics. Agarwal et al. [2] proposed FOBFM model for fast online recommendation learning through effective model initialization using historical information.

[2]This assumption is not always true in reality. For instance, users' preference on items may be influenced by other factors like contexts, e.g., users' emotion and social information [16].

Top-N recommendation

Recently, top-N recommendation that directly generates a list of items for users has attracted a lot of attention. In particular, learning to rank techniques have been applied to extend upon latent factor models for top-N recommendation. CofiRank [24] was the first solution in this trend, where NDCG was used as the loss function, and the model was trained by minimizing over a convex upper bound of the loss function. Kabbur et al. [12] proposed a factored item similarity based method, where a ranking loss function that optimizes the area under the curve (AUC) was used.

Shi et al. proposed a set of ranking-oriented algorithms by fusing learning to rank and latent factor models. TFMAP [20] directly maximized MAP to optimize a ranked list of items for individual users. Tensor factorization was applied to model implicit feedback data (e.g., click) under each type of context. Similar solutions include CLiMF [21] and xCLiMF [19], which focused on implicit binary relevance data and explicit rating respectively.

Topic modeling and collaborative filtering

By fusing topic models and latent factor models, hybrid filtering recommendation methods have been improved. The basic idea is to assign a latent topic to each word of an item, which is represented by a "bag-of-words", and then generate item latent factors by aggregating the topics of all words of this item (i.e., each latent factor is characterized by a topic). A missing rating is predicted by combining the target user's affinity to the topics and the correlation between the topics and the target item. A set of algorithms have been proposed to incorporate topics into user/item latent factors [1, 18, 17] to better model user-item interactions for rating prediction. For instance, collaborative topic regression [22] represented users' latent factors by topic interests and assumed that items' latent factors were generated by a topic model. In [17], a model called "Hidden Factors as Topics" (HFT) was proposed to combine ratings and reviews for product recommendation, where the topics were used as regularizers for the latent factors of users and products.

3. TOP-N RECOMMENDATION FRAMEWORK

In this section, we present our dynamic top-N recommendation framework. We start with framework architecture and notation definition in Section 3.1. A top-N recommendation model, which is built on learning to rank, matrix factorization and LDA, is elaborated in Section 3.2. We propose a heuristic algorithm to combine online and offline recommendations in Section 3.3. Related issues are discussed in Section 3.4.

3.1 Preliminaries

As illustrated in Fig. 1, the proposed recommendation framework consists of two main components: an online component that handles continuously arrived items and an offline component that processes longstanding items. Specif-

ically, the online component stores the coming data in the online storage module (which is typically of small size, compared to the offline storage module) and updates the online recommendation model frequently such that the fresh information can be timely recommended to users. The update frequency depends on characteristics of the coming data (e.g., volume and velocity) as well as the capability of computing infrastructure. Periodically, the data in online storage module is merged to the offline storage module such that the online component only processes the most recent information timely. For instance, the online component only processes a window of 1000 recently entered items (for online recommendation); once the window is full, the 1000 items are moved to offline component and the online component starts processing new items. Due to the large volume of data in the offline storage module, the offline recommendation model is updated less frequently to capture users' preference on the longstanding items. When a recommendation query for a user is issued, both online and offline component provide a recommendation list respectively and the recommendation aggregator serves to combine the two lists based on the user's past rating behavior to provide final top-N recommendation.

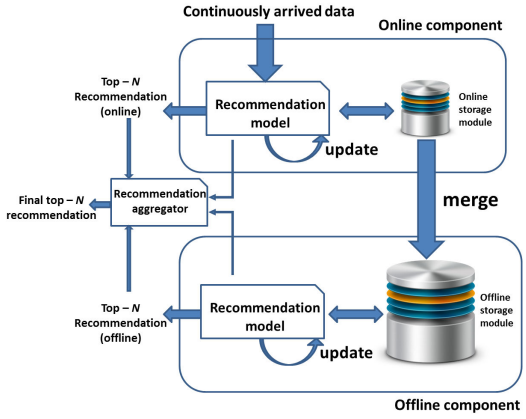


Figure 1: Top-N recommendation framework.

The notations used in this paper are introduced as follows. We denote the user set by $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, and the item set by $\mathcal{V} = \{v_1, v_2, ..., v_n\}$. Any user can rate any item based on her preference. We assume that the value of a rating is a discrete variable in a range $\mathcal{L} = \{L_1, L_2, ..., L_l\}$, e.g., five-point likert scale used by Netflix. A rating provided by user u to item v is denoted by $R_{u,v}$. We assume each item v is associated with a "bag-of-words", denoted by T_v, which is constructed from relevant textual metadata such as the description and tags of the item[3]. The summation of such "bag-of-words" constructs a text corpus $\mathcal{T}$ for LDA.

3.2 Top-N recommendation model

3.2.1 Topic-boosted matrix factorization

Latent factor models, e.g., matrix factorization (MF) have become the state-of-the-art method for recommendation. The rating matrix $R \in \mathcal{R}^{m \times n}$ (m is the number of users and n is the number of items) is factorized into one user-specific matrix $U \in \mathcal{R}^{l \times m}$ and one item-specific matrix $V \in \mathcal{R}^{l \times n}$: $\hat{R} \approx U^T V$, where l is the dimensionality of a latent factor vector that characterizes a user or an item. For user u, the elements of U (i.e., U_u) measure u's affinity to the corresponding latent factors; for item v, the elements of V (i.e., V_v) measure the correlation between v and the corresponding latent factors. Accordingly, the resulting $U_u^T V_v$ captures the correlation between user u and item v, i.e., the predicted rating from u to v. In order to measure the accuracy of rating prediction, an objective function is defined as follows:

$$L = \min_{U,V} \frac{1}{2} \sum_{u=1}^{m} \sum_{v=1}^{n} I_{u,v} (R_{u,v} - U_u^T V_v)^2 + \frac{\lambda}{2} (\| U \|_F^2 + \| V \|_F^2), \quad (1)$$

where $I_{u,v}$ is 1 if user u has rated item v, and 0 otherwise. To avoid overfitting, regularization terms are added, where the parameter λ controls the extent of regularization. Equation 1 can be solved by stochastic gradient descent (SGD), which iteratively updates user-specific and item-specific latent factors to minimize the sum-of-squared-error.

Traditional approaches recommend items to users based on the predicted ratings. However, it is non-trivial to accurately interpret and infer a user's preference simply using ratings, in particular, when such rating information is sparse, which is common in most recommender systems. Recently, some effort has been made to integrate textual contents via topic modeling, e.g., LDA into MF to improve recommendation accuracy [22, 17]. However, in most existing solutions, the dimensionality of latent factors in MF is the same with the number of topics in LDA, and each latent factor is initialized or bounded by the corresponding topic distribution. This implicitly assumes that the latent factors of users and items are completely characterized by textual contents, which is not always true in reality. For instance, a user's preference may be affected by other factors like social relationships and users' emotions [16]. We therefore propose a novel model to fuse MF and LDA by relaxing the arguably strong coupling between topics and items' latent factors.

We apply LDA on the text corpus $\mathcal{T}$ to extract k topics $D = \{d_1, ..., d_k\}$ such that each word w has a probability $\phi_{w,d}$ of being assigned to topic d, and each item v is represented by a topic distribution θ_v. Different from previous methods [22, 17] that reconcile the number of topics and the dimensionality of user/item latent factor vector, in our approach, the topic number k is not necessarily equal to the factor latent vector dimensionality l, and each topic d is also associated with and characterized by a latent factor vector X_d that is sharing the same dimensional space with user/item latent factors.

Accordingly, the correlation between a user and an item, which was captured through the user specific and item specific latent factors, is reformulated by incorporating the effect of topics. Specifically, item specific latent factor matrix V is represented by the product of two matrices:

$$V = XY, \quad (2)$$

where $X \in \mathcal{R}^{l \times k}$ is a topic specific latent factor matrix where each column is a latent factor vector that characterizes the corresponding topic. $Y \in \mathcal{R}^{k \times n}$ is a matrix that stores correlations between items and topics, where the correlation between topic d and item v is measured by the corresponding topic distribution $\theta_{d,v}$ over this item. Therefore, when inferring user u's preference on item v, u's preferences on the k topics are firstly measured through user specific and topic

[3]This assumption holds in most online applications like e-commerce, review sites, social media, etc.

specific latent factors (i.e., dot product), and then the correlations between the k topics and item v are derived, finally the two aspects are fused to capture u's preference on item v. With topics, interactions between users and items can be more meaningfully modeled than traditional scenario where only rating information (less interpretable) is used. Furthermore, the dimensionality of latent factor vector is not bounded by the number of topics thus is more flexible to capture other latent effects besides texts. By incorporating topics the rating from user u to item v can be obtained by

$$\hat{R}_{u,v} \approx U_u^T X Y_v. \tag{3}$$

3.2.2 Learning to rank for recommendation

In contrast to the state-of-the-art approaches that focus on improving the accuracy of rating prediction, we aim to provide top-N recommendation which is more realistic in real-world recommendation scenarios. To this end, we apply learning to rank technique to directly optimize a ranking measure for recommendation. Among numerous loss functions employed by learning to rank algorithms, we choose the one used in RankNet [5], which is not only popular in academia, but is also used in a commercial search engine. Specifically, we employ a pairwise approach to learn top-N item ranking, where the ranking task is formulated as a pairwise classification task, i.e., to rank a pair of items. Note that besides pairwise approach, another two alternatives for learning to rank are pointwise and listwise approaches. Pointwise approach is actually equivalent to rating prediction. For listwise approach, when training the recommendation model, it is non-trial to meaningfully construct a list of ranked item (i.e., items are rated by a user individually thus cannot intuitively construct a list structure). We therefore choose a pairwise approach which can better structure each user's rated items for learning at a finer granularity.

To provide top-N recommendation, existing pairwise based learning to rank models are learned by comparing the relevance of *any* pair of rated items of each user [9]. However, this strategy fails to distinguish characteristics of items thus missing the semantic comparability among items. For instance, it is useful to learn that a user prefers rock music to lyric music, but meaningless to compare rock music and programming books. To solve this issue, we propose to cluster a user's ratings by leveraging LDA such that the rated items can be more meaningfully compared.

Recall that we extract k topics from corpus $\mathcal{T}$, and each item v is represented by a topic distribution θ_v. Based on such topic distributions, we divide user u's rated items into k groups by applying k-mean clustering algorithm[4]. Note that if the size of a group is smaller than a threshold (e.g., 2), this group is merged into the nearest one. *Within* each group, pairs of items, where in each pair the first item is assumed to be ranked higher than the second one (based on ratings), are obtained, and we denote $\Omega_u = \{(v_i, v_j) | v_i \in \mathcal{V}_u \bigwedge v_j \in \mathcal{V}_u \bigwedge v_i \triangleright v_j\}$ as a set of item pais of user u across all groups. Aggregating all users' such item pairs constructs training data for pairwise learning to rank.

It is worth noting that a byproduct of such a topic based clustering strategy is that it removes the item pairs where the comparison of the two items is meaningless (instead of considering all possible item pairs, which is adopted by most existing approaches), thus reducing the volume of training data while keeping its informativeness.

Following its definition in RankNet [5], we apply a cross entropy cost function w.r.t. user u as follows:

$$l^u_{v_i,v_j} = -\bar{P^u_{i,j}} \cdot log P^u_{i,j} - (1 - \bar{P^u_{i,j}}) \cdot log(1 - log P^u_{i,j}), \tag{4}$$

where $P^u_{i,j}$ is the inferred posterior probability (by a logistic function) that item v_i is ranked higher than item v_j by user u:

$$P^u_{i,j} = \frac{e^{f(u,v_i)-f(u,v_j)}}{1 + e^{f(u,v_i)-f(u,v_j)}}, \tag{5}$$

where $f(u, v_i)$ returns item v_i's relevance score, which is the predicted rating from user u to item v_i, obtained by our topic-boosted approach (see Equation 3). $\bar{P^u_{i,j}}$ is the desired target value for the posterior probability (by a logistic function), which is derived using the *known* ratings $R_{u,i}$ and $R_{u,j}$ provided by user u to the pair of items v_i and v_j:

$$\bar{P^u_{i,j}} = \frac{e^{R_{u,i}-R_{u,j}}}{1 + e^{R_{u,i}-R_{u,j}}}. \tag{6}$$

By substituting $P^u_{i,j}$ and $\bar{P^u_{i,j}}$ with Eq. 5 and 6, the loss function Eq. 4 is reformulated as:

$$\begin{aligned} l^{u'}_{v_i,v_j} = -\frac{e^{R_{u,i}-R_{u,j}}}{1 + e^{R_{u,i}-R_{u,j}}} \cdot (f(u, v_i) - f(u, v_j)) + \\ log(1 + e^{f(u,v_i)-f(u,v_j)}). \end{aligned} \tag{7}$$

By considering all users' item pairs, we define the objective function that aims to minimize pairwise loss, taking into account the regularization for avoiding overfitting:

$$\begin{aligned} L' = \min_{U,X} \sum_{u=1}^{m} \sum_{(v_i,v_j)\in\Omega_u} -\frac{e^{R_{u,i}-R_{u,j}}}{1 + e^{R_{u,i}-R_{u,j}}} \cdot (f(u, v_i) - f(u, v_j)) \\ + log(1 + e^{f(u,v_i)-f(u,v_j)}) \\ + \frac{\lambda}{2}(\| U \|^2_F + \| X \|^2_F), \end{aligned} \tag{8}$$

To solve the objective function, which is non-convex, we apply Stochastic Gradient Descent (SGD) to try to find a local minima. We perform gradient descent with respect to user specific and topic specific latent factors respectively:

$$\begin{aligned} \frac{\partial L'}{U_u} = \sum_{(v_i,v_j)\in\Omega_u} -P^u_{i,j}(XY_i - XY_j) \\ + \frac{e^{U_u XY_i - U_u XY_j}(XY_i - XY_j)}{1 + e^{U_u XY_i - U_u XY_j}} \\ + \lambda U_u. \end{aligned} \tag{9}$$

$$\begin{aligned} \frac{\partial L'}{X} = \sum_{u=1}^{m} \sum_{(v_i,v_j)\in\Omega_u} -P^u_{i,j}(U_u Y_i^T - U_u Y_j^T) \\ + \frac{e^{U_u XY_i - U_u XY_j}(U_u Y_i^T - U_u Y_j^T)}{1 + e^{U_u XY_i - U_u XY_j}} \\ + \lambda X. \end{aligned} \tag{10}$$

The latent factors are updated iteratively:

$$U_u \leftarrow U_u - \gamma \frac{\partial L'}{U_u}, \quad X \leftarrow X - \gamma \frac{\partial L'}{X}, \tag{11}$$

[4]K-means clustering algorithm is applied due to its simplicity. More sophisticated clustering algorithms may be used, but the discussion on the tradeoff between the improved clustering quality and the increased computational complexity is beyond the scope of this work.

where γ is the learning rate. By sorting the predicted ratings (i.e., relevance scores) of items (in descending order) that user u has not rated, we generate top-N items for user u.

3.3 Combining online and offline recommendation

To produce final top-N recommendation by combining recommendations from online and offline components[5], we investigate a user's past rating behavior to assess her preference on the freshness of items. We denote the freshness of an item v when it was rated by ψ_v, which is a binary variable where 1 indicates that this item was fresh (i.e., processed by the online component) and 0 indicates that this item was longstanding (i.e., processed by the offline component).

For user u, we record the freshness of each item that u has rated: $\{\psi_1, \psi_2, ...\}$. These freshness indicators are modeled as observations of independent Bernoulli trials. In each trial, the success probability (i.e., the probability of rating a fresh item) is modeled by Beta distribution with parameters α and β (we start with $\alpha = \beta = 1$, which translates into complete uncertainty about the distribution of the parameter, modeled by uniform distribution: Beta(1, 1) = U(0, 1)). After observing s successes in n trials, the posterior density of the probability is Beta($\alpha = \alpha + s$, $\beta = \beta + n - s$). The expectation probability[6] of rating a fresh item is then obtained by $\Psi_u = \frac{\alpha}{\alpha+\beta}$. The higher the Ψ_u, the more likely user u will rate a fresh item.

Given the online recommendation list Ω^u_{on} and offline recommendation list Ω^u_{off} for user u, as well as u's preference on the freshness of items Ψ_u, the final top-N recommendation list Ω^u is generated iteratively: at each iteration, the top fresh item v' is picked from Ω^u_{on} with a probability of Ψ_u; otherwise, the top longstanding item v'' is picked from Ω^u_{off}. Once chosen, the item v' or v'' is appended to the end of Ω^u, and removed from Ω^u_{on} or Ω^u_{off}. This process continues until the size of Ω_u increases to N.

3.4 Discussion

To achieve personalized top-N recommendation, we proposed a recommendation framework by leveraging learning to rank, matrix factorization and LDA. Two important issues, scalability and cold-start, are discussed as follows.

The time complexity of the optimization procedure is $O(m \cdot \tilde{t}^2)$, where m is the number of users, and $\tilde{t}$ is the average number of items per user [5]. In practice, sampling methods might be applied to sample a subset of items for each users (i.e., decreasing the value of $\tilde{t}$), and hence reduce the computational complexity. Furthermore, for each user, we cluster the rated items based on their topic distribution to make the pairwise item relevance comparison more meaningful[7]. This greatly reduces the number of comparisons (i.e., $\tilde{t}^2$), thus simplifying the model training. Another way to cope with the complexity of the model is to parallelize the optimization procedure, e.g., distributed SGD [11] or downpour SGD [8], which makes our approach applicable to even larger data.

It is also worth noting that "cold-start" is still an open question in most recommender systems. For our approach, this issue is particularly important when fresh items are merged from online storage module to offline storage module (i.e., the input data for online recommendation model will be sparse). To address this issue, each user's most recently rated items are not only merged to the offline storage module but also kept in the online storage module as the buffer to bootstrap newly arrived items[8], e.g., by measuring the similarity between two items based on their topic distributions. Moreover, a user's recent ratings reflect her current preference, so the derived online recommendation is expected to be more accurate. Regarding the completely new users, existing solutions (e.g., [23]) can be applied on the basis of our recommendation approach, however, the detailed discussion about this issue is beyond the scope of this paper.

4. EVALUATION

4.1 Methodology

We use a dataset collected from Douban (www.douban.com), one of the largest Chinese based social platforms for sharing reviews and recommendations for books, movies and music. Users provide ratings in 5-star scale to indicate their preference on items. The dataset contains ratings submitted from 2007 to 2011. Table 1 summarizes the statistics of Douban dataset (shared by the first author of [26]).

Table 1: Statistics of Douban data

	# of ratings	# of users	# of items
Book	1,097,148	33,523	381,767
Movie	2,828,585	33,561	87,081
Music	1,387,216	29,287	257,288
All	5,312,949	36,673	726,136

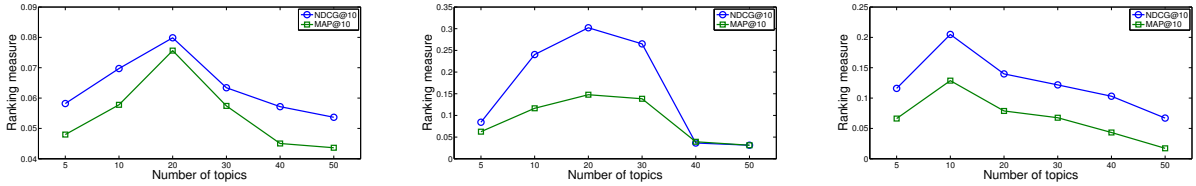
Besides rating information, we crawled each item's textual metadata such as the summary and tags assigned by users for topic modeling. For our approach, we keep the items that enter the system before 2011 in offline storage module (see Fig. 1), and treat the items that enter the system in 2011 as continuously arrived data which will be handled by online component. The online recommendation model is updated when every 1000 new items enter the system and the offline recommendation model is updated when every 10,000 new items are merged from online storage module to offline storage module (i.e., the online model is updated 10 times more frequently than the offline model). Note that the ratings submitted in 2011 will be used as test data, and the ones submitted before 2011 will be used as training data. Since top-N recommendation is provided by combining online recommendation and offline recommendation, so the two components cannot be evaluated separately. Besides Douban data, we also use the Delicious bookmarks data [6]. In this dataset, 1,867 users bookmarked 69,226 URLs with 53,388 tags (for topic modeling). In the experiments, the purpose of a recommendation model is to recommend URLs that are likely to be bookmarked by users (i.e., binary rating is used). Note that due to space limitation, Delicious dataset is only used in comparison study in Section 4.2.2.

[5]Remind that online and offline components use the same recommendation model elaborated in Section 3.2.

[6]A straightforward extension is to add a decay factor to emphasize users' recent preference on items' freshness to improve prediction accuracy. We leave as a future work a more detailed discussion on such extensions.

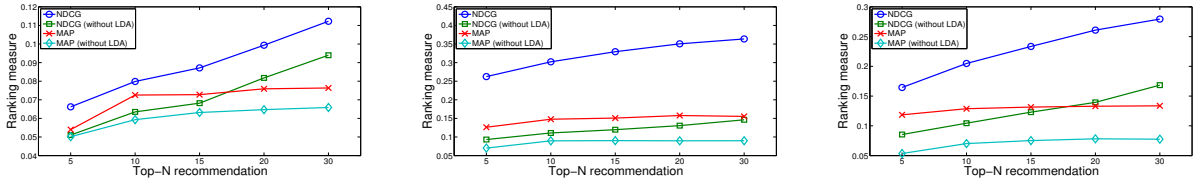
[7]This may worsen the issue of data sparsity. In practice, we only apply clustering strategy to users who have rated sufficient number of items (e.g., at least 20)

[8]The number of users' most recently rated items that are kept in online component is a design parameter, which can be configured according to characteristics of the system or by cross validation. We set this number to 1 in our experiments.



(a) Book data. (b) Movie data. (c) Music data.

Figure 2: Performance of our approach with varying topic number (top-10 recommendation, Douban data).



(a) Book data. (b) Movie data. (c) Music data.

Figure 3: Influence of textual content (Douban data).

We compare our approach with several representative baselines: (1) **BasicMF**. This is the basic matrix factorization model. Top-N recommendation is provided by sorting the candidate items in descending order of the predicted ratings. The dimensionality of factors, regularization parameter and learning rate are set to 10, 0.1 and 0.01 respectively (by cross validation). (2) **RMFX** [10]. This approach uses a (hinge loss based) pairwise approach with matrix factorization to optimize the ranking for social streams like Twitter. When processing continuously arrived items, it applies random sampling with a reservoir to sample user-item pairs for model update. The dimensionality of factors, regularization parameter, learning rate, and reservoir size are set to 10, 0.1, 0.1 and 100 respectively (by cross validation). (3) **CLiMF** [21]. This is a ranking-oriented algorithm that directly optimize a ranking measure Mean Reciprocal Rank (MRR). The dimensionality of factors, regularization parameter and learning rate are set to 10, 0.001 and 0.001 respectively (by cross validation). The implementation of this model is publicly available at http://dmirlab.tudelft.nl/users/yueshi. (4) **timeSVD++**. This is a matrix factorization model incorporating temporal dynamics proposed in [13]. User bias, item bias and user latent factors are modeled as functions of time. For each of these approaches, we use the ratings submitted from 2007 to 2010 as the training data to train the model and validate its performance using ratings submitted in 2011. To obtain time-aware recommendation, we updated these models (except RMFX) when every 5000 new items enter the system. 5000 is chosen as the compromise of the online and offline components of our approach (see previous paragraph for the settings of our approach). For RMFX, the model is updated when every 1000 items enter the system. This is to compare with our approach's online component that is particularly designed to handle fresh items. Experiments are conducted 10 times and the averaged results are presented for each approach. Note that error bars are omitted in figures to avoid cluttering but in Tab. 2, we provide 95% confidence interval (all comparison results are statistically significant).

We measure the performance of recommendation models using Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG), two well-known metrics for measuring the performance of ranking algorithms.

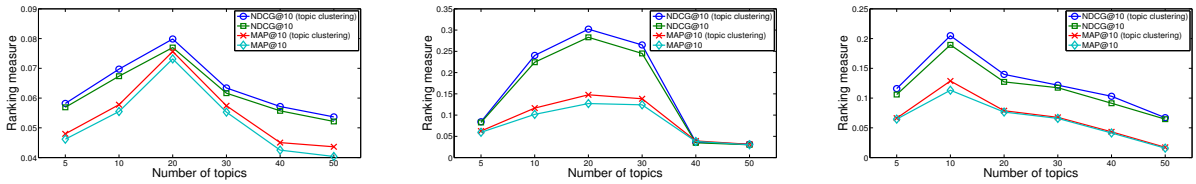
4.2 Results

4.2.1 *Influence of design parameters*

As argued in Section 3.2.1, the number of topics in LDA should not be tied to the dimensionality of latent factors in matrix factorization such that textual contents can be more flexibly incorporated to model interactions between users and items. We thus first demonstrate the performance of our approach when different number of topics is set. Note that the dimensionality of latent factors, regularization parameter and learning rate are set to 5, 0.001 and 0.00001 which are optimized by cross validation. From Fig. 2 we observe the performance of our approach varies a lot with different number of topics. For every type of item (i.e., book, movie and music), the general trends are both NDCG and MAP first increase, when arriving at a certain point, they start decreasing with the increasing number of topics. Specifically, for book data and movie data, 20 topics achieve the highest NDCG and MAP, while for music data, the optimal number of topics is 10. This observation supports our claim that topic number should not be tied to the dimensionality of latent factors (i.e., 5 in this case), and a suitable topic number significantly improves the performance of our approach.

We next validate the effectiveness of combining LDA and matrix factorization for modeling users' preference on items for ranking. Fig. 3 shows the comparison results (with different recommendation list size) of our approach and a variant where textual contents are not considered. It is clear that when both textual contents and rating information are used, NDCG and MAP are significantly improved. This is because by deriving topic distribution for each item, users' preference can be more accurately modeled, especially when the target user and/or item's ratings are sparse (i.e., a single piece of textual content can reveal many of an item's characteristics thus is more informative than a single rating).

Recall that one contribution of our work is to cluster a user's rated items based on their topic distributions such that the training data (i.e., item pairs for optimizing pairwise loss function) is more compact and more meaningful. We next demonstrate the effectiveness of such item clustering. From Fig. 4 we notice that in all cases, considering topic based clustering evidently improves the performance of a variant of our approach where no clustering is conducted. This demonstrates that removing unnecessary item



(a) Book data. (b) Movie data. (c) Music data.

Figure 4: Influence of topic clustering (top-10 recommendation, Douban data).

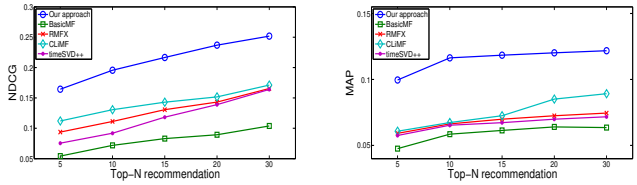
Table 2: Performance comparison (top-10 recommendation, 95% confidence interval, Douban dataset)

	Book		Movie		Music	
	NDCG@10 (95% C.I.)	MAP@10 (95% C.I.)	NDCG@10 (95% C.I.)	MAP@10 (95% C.I.)	NDCG@10 (95% C.I.)	MAP@10 (95% C.I.)
Our approach	**0.0799 (0.0785,0.0812)**	**0.0726 (0.0711,0.0740)**	**0.3021 (0.3001,0.3046)**	**0.1477 (0.1455,0.1498)**	**0.2048 (0.2014,0.2080)**	**0.1287 (0.1265,0.1306)**
BasicMF	0.0266 (0.0250,0.0281)	0.0255 (0.0238,0.0275)	0.0851 (0.0840,0.0863)	0.0754 (0.0741,0.0769)	0.0837 (0.0815,0.0861)	0.0416 (0.0399,0.0435)
RMFX	0.0525 (0.0504,0.0551)	0.0486 (0.0452,0.0509)	0.1923 (0.1895,0.1955)	0.0822 (0.0798,0.0842)	0.0985 (0.0960,0.0911)	0.0520 (0.0505,0.0538)
CLiMF	0.0585 (0.0565,0.0604)	0.0503 (0.0481,0.0525)	0.2152 (0.2111,0.2198)	0.0967 (0.0915,0.1008)	0.1180 (0.1150,0.1206)	0.0600 (0.0573,0.0631)
timeSVD++	0.0412 (0.0397,0.0431)	0.0395 (0.0371,0.0418)	0.1634 (0.1611,0.1660)	0.0951 (0.0921,0.0983)	0.1205 (0.1188,0.1224)	0.0614 (0.0588,0.0642)

pairs indeed refines the training data thus improving ranking measures. On average, such a clustering strategy improves the performance of our approach in terms of NDCG/MAP, by 2.99%/4.89%, 5.37%/9.02% and 7.99%/5.92% for book data, movie data and music data respectively.

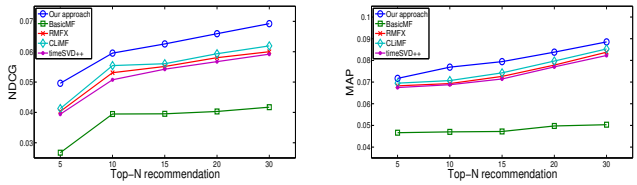
4.2.2 *Comparison study*

We compare the performance of our approach with that of the state-of-the-art approaches (see Section 4.1). Note that



(a) NDCG. (b) MAP.

Figure 5: Performance comparison (Douban data).



(a) NDCG. (b) MAP.

Figure 6: Performance comparison (Delicious data).

for our approach, optimal topic number is applied for different type of items. Table 2 summarizes NDCG and MAP for all recommendation approaches when top-10 recommendation is provided. Not surprisingly, BasicMF, which simply predicts a user's rating to an item for ranking, produces the lowest NDCG and MAP. This shows that optimizing rating prediction does not necessarily generates good recommendation ranking, in particular, when rating information is sparse. As a ranking-oriented recommendation approach, CLiMF significantly outperforms BasicMF. This proves the effectiveness of learning to rank for top-N recommendation. By optimizing a ranking measure, CLiMF directly generates a recommendation list instead of focusing on improving rating prediction accuracy. Similarly, RMFX, which applies pairwise learning to rank also outperforms BaseMF. However, to better handle scalability issue, RMFX applies sampling techniques to process the incoming items, which inevitably lose information (and hence, the accuracy). This makes RMFX slightly less accurate than CLiMF. timeSVD++, although is not particularly designed for ranking, by incorporating temporal effects into matrix factorization (via time-aware user/item bias and latent factors), still generates reasonable results: for music data, timeSVD++ is even better than CLiMF and RMFX. In all cases, our model consistently outperforms other methods due to four key designs: (1) learning to rank is applied to directly optimize item ranking instead of rating prediction; (2) textual contents are incorporated (by fusing LDA and matrix factorization) to better model users' preference; (3) topic distribution based item clustering refines the training data to more efficiently fit the model; (4) new items and longstanding items are modeled separately for providing personalized recommendation based on individual users' preference on the freshness of items.

We also compare the performance of all recommendation approaches when recommendation list size varies. Fig. 5(a) and 5(b) summarize NDCG and MAP for each recommendation approach when top-5, 10, 15, 20, 30 recommendations are provided. Note that we mix book data, movie data and music data to conduct comprehensive comparison study. Similar to previous comparison results, our approach generates the highest NDCG and MAP. In summary, by averaging performance when different recommendation list size is applied, our approach improves BasicMF, RMFX, CLiMF and timeSVD++ by 168.98%, 72.40%, 50.27% and 87.64% in terms of NDCG, 96.73%, 72.83%, 61.21% and 75.95% in terms of MAP. The comparison results using Delicious data (see Fig. 6) demonstrate the similar trends.

5. CONCLUSION

In this paper, we present a dynamic recommendation framework that consists of an online component that quickly pro-

cesses the continuously arrived items and an offline component that captures the influence of longstanding items. We apply learning to rank to optimize a pairwise loss function. LDA is integrated to combine rating information and textual contents to better model users' preference on items. Furthermore, topic distribution based item clustering refines training data. A heuristic algorithm is proposed to combine online and offline recommendations based on users' preference on the freshness of items. Two real datasets based experiments demonstrate that our approach significantly outperforms the state-of-the-art models.

As for the future work, we intend to explore more sophisticated mechanisms to derive users' preference on the freshness of items for recommendation. Another direction is to investigate to further reduce computational complexity to better handle scalability issue (e.g., by sampling).

6. ACKNOWLEDGEMENTS

This work was partially supported by the grant Reconcile: Robust Online Credibility Evaluation of Web Content from Switzerland through the Swiss Contribution to the enlarged European Union.

7. REFERENCES

[1] D. Agarwal and B.-C. Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM WSDM*, 2010.

[2] D. Agarwal, B.-C. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *Proceedings of the 16th ACM SIGKDD*, 2010.

[3] M. Aly, S. Pandey, V. Josifovski, and K. Punera. Towards a robust modeling of temporal interest change patterns for behavioral targeting. In *Proceedings of the 22nd WWW*, 2013.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*, 2005.

[6] I. Cantador, P. Brusilovsky, and T. Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec). In *Proceedings of the 5th ACM RecSys*, 2011.

[7] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu. Collaborative personalized tweet recommendation. In *Proceedings of the 35th ACM SIGIR*, 2012.

[8] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep networks. In *Proceedings of the 25th NIPS*, 2012.

[9] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *Proceedings of the 6th ACM RecSys*, 2012.

[10] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *Proceedings of the 6th ACM RecSys*, 2012.

[11] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD*, 2011.

[12] S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD*, 2013.

[13] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD*, 2009.

[14] N. Lathia, S. Hailes, and L. Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd ACM SIGIR*, 2009.

[15] N. N. Liu, M. Zhao, E. Xiang, and Q. Yang. Online evolutionary collaborative filtering. In *Proceedings of the 4th ACM RecSys*, 2010.

[16] X. Liu and K. Aberer. Soco: a social network aided context-aware recommender system. In *Proceedings of the 22nd WWW*, 2013.

[17] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM RecSys*, 2013.

[18] S. Purushotham and Y. Liu. Collaborative topic regression with social matrix factorization for recommendation systems. In *Proceedings of the 29th ICML*, 2012.

[19] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. xclimf: Optimizing expected reciprocal rank for data with multiple levels of relevance. In *Proceedings of the 7th ACM RecSys*, 2013.

[20] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th ACM SIGIR*, 2012.

[21] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM RecSys*, 2012.

[22] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD*, 2011.

[23] F. Wang, W. Pan, and L. Chen. Recommendation for new users with partial preferences by integrating product reviews with static specifications. In *Proceedings of 21st UMAP*, 2011.

[24] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofirank - maximum margin matrix factorization for collaborative ranking. In *Proceedings of the 21st NIPS*, 2007.

[25] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of SDM*, 2010.

[26] E. Zhong, W. Fan, J. Wang, L. Xiao, and Y. Li. Comsoc: adaptive transfer of user behaviors over composite social network. In *Proceedings of the 18th ACM SIGKDD*, 2012.