User Recommendation in Content Curation Platforms

Jianling Wang, Ziwei Zhu and James Caverlee
Department of Computer Science and Engineering, Texas A&M University
{jlwang,zhuziwei,caverlee}@tamu.edu

ABSTRACT

We propose a personalized user recommendation framework for content curation platforms that models preferences for both users and the items they engage with simultaneously. In this way, user preferences for specific item types (e.g., fantasy novels) can be balanced with user specialties (e.g., reviewing novels with strong female protagonists). In particular, the proposed model has three unique characteristics: (i) it simultaneously learns both user-item and user-user preferences through a multi-aspect autoencoder model; (ii) it fuses the latent representations of user preferences on users and items to construct shared factors through an adversarial framework; and (iii) it incorporates an attention layer to produce weighted aggregations of different latent representations, leading to improved personalized recommendation of users and items. Through experiments against state-of-the-art models, we find the proposed framework leads to a 18.43% (Goodreads) and 6.14% (Spotify) improvement in top-k user recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**;

KEYWORDS

Recommendation; Content Curation; Attentive Adversarial Model

ACM Reference Format:
Jianling Wang, Ziwei Zhu and James Caverlee. 2020. User Recommendation in Content Curation Platforms. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371822

1 INTRODUCTION

Content curation platforms are powered by users who surface interesting content – via reviews, pins, boards, ratings and other actions. For example, Pinterest users curate and comment on links or images to form thematic "boards" on various topics like Recipes and Fashion. Similarly, Goodreads provides a platform for users to curate interesting books via tagging, ratings, and reviews. And music streaming services like Spotify allow users to create and share playlists containing combinations of individual tracks. Unlike content *creators* in social media who generate new digital artifacts such as tweets, blog posts, or photos, users in curation platforms, who we refer to as *curators*, collect and organize existing content, while often providing additional insights via comments, reviews, or ratings [11, 21]. In this way, these *curators* can provide a human-powered overlay that can link seemingly un-related items (e.g., a collection of books that are thematically related though from different genres).

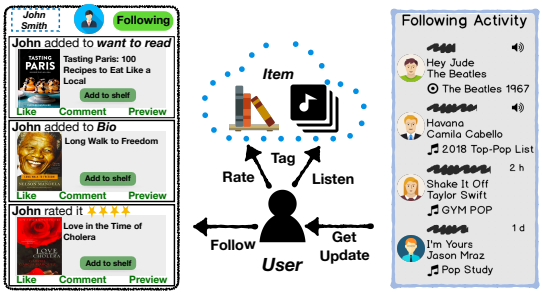

Figure 1: In curation platforms, users acting as *curators*, collect and organize existing content via tagging, reviews, or ratings. By receiving updates from whom they follow, users can be exposed to interesting items and curation decisions.

By connecting to curators, users can discover new items, new collections, and new connections among items built via human (rather than algorithmic) power. For example, as illustrated in Figure 1, a Spotify user can follow other users and thus keep track of their listening activities (like songs, albums, and playlists). In Goodreads, users will see updates including new bookshelves, ratings and text reviews from those whom they follow. Indeed, around 50% of Spotify's 100 million users listen to human-curated playlists [37] and researchers have shown how the power of human curation can serve as a significant component of modern recommender systems to connect users to items [16, 24].

But how can we recommend interesting curators to follow? This challenge of *user recommendation in content curation platforms* is vitally important and yet most existing methods that rely on traditional item-level recommendation [2, 3, 9, 35, 38] or on expert-finding approaches [8, 12, 26, 29] may not capture the important relationships among both curators and the items they curate. Users in curation platforms are complex amalgamations of the specific items they interact with (e.g., sci-fi books), their style of curation (e.g., reflecting personal interests), and the complex interactions between items and the curated lists themselves. For example, a specific item type like a fantasy novel may be curated by someone with an interest in novels with strong female protagonists, while the same novel may be curated by someone else with an interest in space warfare. Compared to previous works on friend recommendation [10, 32], topical user recommendation (e.g., finding other users interested in fashion) [14, 30], and expert finding [8, 12, 26, 29], there is a research gap in curator recommendation that carefully

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371822

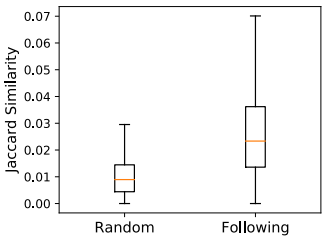

Figure 2: Similarity between random user pairs and user pairs where the one follows the other in Goodreads. (Each user is represented with a binary vector over all books, in which a "1" indicates that the user has curated the book by adding it to personal bookshelf, leaving comment or rating.)

balances user preferences for specific item types and preferences for different curator specialties.

Hence, we are motivated to develop a new model for *curator recommendation* that leverages this linkage between user-curator following relationships and the items they are interested in. Ultimately, this work aims to provide users with *improved personalized recommendation on who to follow (the primary task) and interesting items (the supplementary task) simultaneously.* Such an effort requires careful modeling of these two aspects – curator preferences and item preferences – in a unified framework. Curator recommendation faces challenges of extreme sparsity in many platforms, where there is a long-tail of users with limited feedback (say, by following). The densities of feedback on users and items are usually unbalanced in content curation platforms. Though the feedback on items can enrich user profiling in curator recommendation, directly fusing user preferences on different aspects can introduce noise during the training process and lead to bad performance for both tasks. Furthermore, since the preferences for curators and items can vary by individual users, such an approach should take care to provide personalization for individual users.

Towards tackling these challenges, we propose the (*CuRe*) framework for content **Cu**ration platform (*CuRe*) user **Re**commendation. In particular, the proposed *CuRe* has three unique characteristics:

- First, we propose to effectively extract and represent the preferences of users on curators and items through a multi-aspect autoencoder model that simultaneously learns both user-curator and user-item preferences.
- Second, we develop an adversarial framework to fuse the latent representations of user preferences on curators and items to construct shared factors. In this way, CuRe tackles the primary task of curator recommendation, while also supporting the supplementary task of item recommendation.
- Third, to capture personal preferences of different users, we incorporate an attention layer to produce weighted aggregations of different latent representations, leading to improved personalized recommendation of curators, with the added benefit of high-quality item recommendation.

With experiments on two platforms (Goodreads and Spotify) supporting different kinds of curation behaviors, we find that the proposed model can provide improved curator recommendation. Comparing with a suite of baselines from simple models to recently introduced state-of-the art models, we find that CuRe leads to an 18.43% and 6.14% improvement in top-k curator recommendation for Goodreads and Spotify, and a 27.38% and 9.61% improvement in the cold-start scenarios.

2 RELATED WORK

Content Curation Platforms and User Recommendation. Recently there have been efforts to study curation platforms. In [5, 25], the authors recommend user-generated lists based on the occurrences of items among lists and interactions of users on items. Work in [18, 44] focuses on "board" recommendation in Pinterest. However, there is a gap in our understanding of finding interesting users to follow. The connections between users and curators are amalgamations of curated contents, individual items and personal style, which pose new challenges compared to traditional friend recommendation in social networks [10, 32]. Somewhat similar to our notion of curator is research on finding expertise to improve search and recommendation, including topical user recommendation [14, 30, 45, 46] and expert finding [8, 12, 26, 29]. For example, UserRec [46] models users' similarity based on the tag-graph and topic distributions, with which it can help in connecting users sharing similar interest. Yan and Zhou [43] propose to improve friend recommendation by capturing the relationships among users, their friends and interest with tensor factorization. However, these and related methods usually rely on users' explicit (topical) tags on experts, or the semantics of tags and content created by users, which are not always available in content curation platforms. In contrast, we focus on curation decisions of these users to model users and the items they curate simultaneously.

Multi-task learning and joint recommendation. CuRe jointly learns user preferences on curators and items akin to multitask learning [6], which has been widely adopted in different recommendation scenarios [2, 5, 19, 27, 34, 40]. By sharing feature representations across different tasks, it enables the recommender system to fully leverage hidden signals from the sparse data through regularization. Kang et al. [19] set item correlation prediction and next interaction prediction as supplementary tasks, to improve the performance of sequential item recommendation. In GRU-MTL [2], they train the recurrent model with two tasks – tag prediction and text recommendation, to ultimately improve the text recommendation. Similarly, in [27, 40], they combine personalized item recommendation with opinionated text content modeling together to achieve better results in the recommendation. Enhanced with transferring knowledge between different social media sites, Crossfire [34] generates recommendations of friends and items at the same time. In contrast, our proposed approach combines user recommendation in curation platforms with the supplementary task of item recommendation to better model user preferences.

3 CURE: CURATOR RECOMMENDATION

In this section, we propose to tackle the problem of curator recommendation through a novel model that aims to balance user-curator interactions with user-item preferences. We begin with the problem statement and then step through the development of our model.

3.1 Problem Setting

Let $\boldsymbol{U} = \{u_1, u_2, ..., u_N\}$ be the set of users in the content curation platform. Users leave implicit feedback on other users, e.g., through the "following" action. We refer to users that are followed by others as *curators*. Hence, the candidate set of *curators* that users can follow is the same as $\boldsymbol{U}$. By following a curator, that user will receive updates from them. Of course, if a user does not follow c, then the user may be uninterested or merely unaware of c. For user u, we use a binary vector $\mathbf{y}_u = [y_{u1}, y_{u2}, ..., y_{uN}]$ to indicate the users followed by u, in which $y_{uc} = 1$ denotes u is following c and $y_{uc} = 0$ means u does not follow c. At the same time, users are able to leave (implicit) feedback on items. Let $\boldsymbol{I} = \{i_1, i_2, ..., i_P\}$ denote the set of items we consider. Similarly, we use binary vector $\mathbf{t}_u$ to indicate items user u has left feedback on.

Since user-curator-item preferences are closely connected in content curation platform, we propose to capture user preferences on individual items and curators simultaneously. Ultimately, our goal is that, $\forall u \in \boldsymbol{U}$, we want to recommend a ranked list of curators to follow $\mathbf{l}_u^C \subset \boldsymbol{U}$ (which we refer to as our *primary task*) and a ranked list of interested items $\mathbf{l}_u^I \subset \boldsymbol{I}$ (which we refer to as our *supplementary task*). This dual approach – considering both a primary and supplementary task – is built around three guiding research questions: (RQ1) How can we represent user's preferences on curators and items simultaneously? (RQ2) How can we integrate these preferences so that user-curator and user-item preferences mutually reinforce each other? (RQ3) How can we personalize the learned latent representations to capture user-specific preferences?

3.2 RQ1: Learning Curator & Item Preferences

Since users leave feedback on both *items* and *curators* who collect and organize items, we first aim to model users based on these two aspects – preference on items and preference on curators. In this section, we start from an autoencoder structure to effectively uncover the latent representations of user-curator preferences. We then extend this basic model to incorporate user-item preferences.

Curator Preference Representation via Autoencoder. As the first step, we adopt a basic autoencoder model to build a user recommender. An autoencoder is a one-hidden layer neural network consisting of an encoder and a decoder, which aims to reconstruct the input from the hidden latent representation. AutoRec [33] demonstrates how the autoencoder can be generalized to collaborative filtering and applied for recommender systems. The Denoising Autoencoder (DAE) [39] extends the basic autoencoder by corrupting the original input vector with additive noise (denoising), resulting in improved performance.

Given the binary vector $\mathbf{y}_u = [y_{u1}, y_{u2}..., y_{uN}]$ representing user u's implicit feedback on all the curators, we can first corrupt this original input with some noise. We mask out each dimension of $\mathbf{y}_u$ with a probability of η and then we get the corrupted vector $\widetilde{\mathbf{y}_u} = [\widetilde{y_{u1}}, \widetilde{y_{u2}}, ..., \widetilde{y_{uN}}]$. We feed this vector into the encoder and generate the hidden latent representation:

$$\mathbf{h}_u = \delta(\mathbf{V}\widetilde{\mathbf{y}_u} + \mathbf{b}), \tag{1}$$

in which $\mathbf{V}$ is the weight matrix for the encoder and $\mathbf{b}$ is the bias term. $\delta(\cdot)$ denotes the Sigmoid function. Then the decoder will reconstruct the input from the hidden latent representation with

$$\widehat{\mathbf{y}_u} = \delta(\mathbf{W}\mathbf{h}_u + \mathbf{b'})$$

where $\mathbf{W}$ and $\mathbf{b'}$ denote the weight matrix and bias of the decoder accordingly. Then the loss is the reconstruction error calculated between the output and the uncorrupted input.

$$L = \sum_{u \in \mathbf{U}} (\sum_{c \in \mathbf{S}_u^C} \widehat{y_{uc}} - y_{uc})^2$$

Practically, while considering the construction loss, besides the users with positive feedback from user u, $\mathbf{S}_u^C$ also contains negative samples of curators for whom u has not left any feedback. With a well-trained DAE model, we can immediately generate personalized curator recommendations for u based on the reconstructed output $\widehat{\mathbf{y}_u}$, in which $\widehat{y_{uc}}$ represents u's preference on c. Such an approach, however, builds a representation $\mathbf{h}_u$ for user u based solely on historic user-curator interactions, without considering the importance of user-item preferences as well.

Joint Curator-Item-DAE. Hence, we next enrich the latent representation $\mathbf{h}_u$ with our supplementary task (item recommendation). From Figure 2, we find that user preferences on items can hint on whom the user wants to follow. To enrich our curator recommendation with this additional information, we can first use a DAE to model user preferences on items (since we have user feedback vectors on items). Let $\mathbf{h}_u^I$ denote the hidden latent factors for user u generated from this supplementary DAE. We supplement $\mathbf{h}_u$ with the user-specific vector $\mathbf{h}_u^I$ from u's feedback on items to improve our curator recommendation.

The intuition of joint Curator-Item-DAE is to recover not only the feedback vectors on curators but also the vectors for user-item interactions [1]. In this case, the primary goal is to recover the curator feedback vector while the supplementary task is to recover the item feedback vector. For clarity, we rename the latent representation $\mathbf{h}_u$ embedded from user-curator interactions (with Equation 1) as $\mathbf{h}_u^C$. Given the supplementary item feedback vector $\mathbf{t}_u$ of user u, we first corrupt them with the mask-out noise (*denoising*). And then encode them into hidden representations $\mathbf{h}_u^I$:

$$\mathbf{h}_u^I = \delta(\mathbf{V}^I\widetilde{\mathbf{t}_u} + \mathbf{b}^I)$$

where $\widetilde{\mathbf{t}_u}$ is the corrupted version of $\mathbf{t}_u$. Then we construct the enriched version of $\mathbf{h}_u$ by integrating the hidden representations $\mathbf{h}_u^I$ and $\mathbf{h}_u^C$ via element-wise addition $\mathbf{h}_u = \mathbf{h}_u^C + \mathbf{h}_u^I$.

We feed the combined hidden representation $\mathbf{h}_u$ of user u into separate decoders for the primary task and supplementary task.

$$\widehat{\mathbf{y}_u} = \delta(\mathbf{W}\mathbf{h}_u + \mathbf{b'}) \qquad \widehat{\mathbf{t}_u} = \delta(\mathbf{W}^I\mathbf{h}_u + \mathbf{b'}^I)$$

Then the loss function is the weighted sum of reconstruction loss for the primary task (curator recommendation) and the supplementary task (item recommendation):

$$L_{rec} = \sum_{u \in \mathbf{U}} (\sum_{c \in \mathbf{S}_u^C} (\widehat{y_{uc}} - y_{uc})^2 + \alpha \sum_{i \in \mathbf{S}_u^I} (\widehat{t_{ui}} - t_{ui})^2), \tag{2}$$

where $\mathbf{S}_u^I$ includes all the items u has left feedback on (positive) and a subset of items u hasn't left feedback on (negative samples). We use α to adjust the weight for the supplementary task.

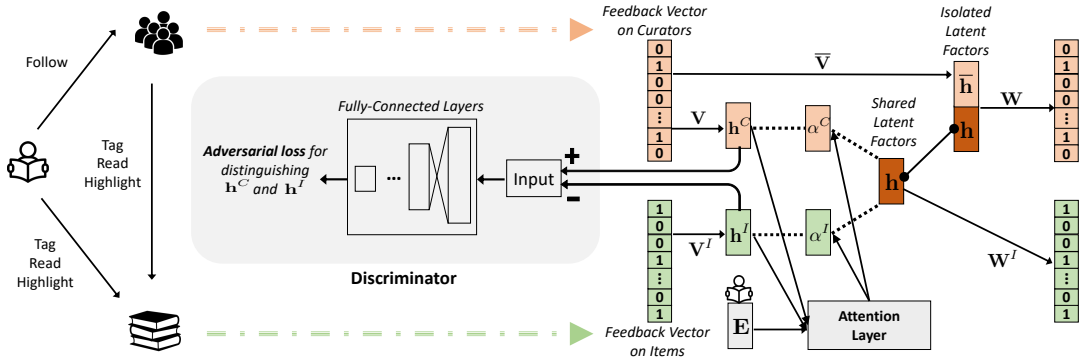

Figure 3: The structure of CuRe. The model generates user's latent presentations with feedback vectors on curators and items. While fusing those representations, to force $\mathbf{h}_u^C$ and $\mathbf{h}_u^I$ to live in a shared space, the *Discriminator* is trained to discriminatively predict whether a latent representation is from $\mathbf{h}^I$ or $\mathbf{h}^C$. The attention layer can generate user-specific weighted aggregations $\mathbf{h}$ for user's preferences on curators and items.

3.3 RQ2: Fusing Latent Representations

In our approach so far, the combined representation $\mathbf{h}$ – which is the element-wise aggregation of samples in $\mathbf{h}^I$ and samples in $\mathbf{h}^C$ – is designed to contain shared information for both tasks (curator recommendation and item recommendation). Such a shared representation would require that the element in the same dimension of $\mathbf{h}^I$ and $\mathbf{h}^C$ should correspond to the same latent factor of users while forming the shared representation. That is, we would like samples in $\mathbf{h}^I$ and $\mathbf{h}^C$ to have similar distributions and so share the same space. By doing so, we can improve the effectiveness of the element-wise fusion for latent representations from different aspects. Furthermore, we find empirically that such a fused space leads to faster convergence, more robustness to noise in user feedback, and higher-quality recommendation. Thus we adopt an adversarial learning framework to fuse the latent representations from the curator and item aspects before aggregating them.

There are prior works on adapting variational autoencoder (VAE) to collaborative filtering, in which the variational encoder tries to approximate a standard user-agnostic Gaussian prior (distribution) [22] or user-dependent priors learned from users' preference in text reviews [20]. Recently, adversarial autoencoders [28] have been proposed and shown to perform better than VAE. Following the idea of generative adversarial networks (GAN), adversarial encoders deploy the adversarial loss for distinguishing samples from hidden layers in an autoencoder and samples from the desired distribution. As in the work of [23], we supplement the reconstruction loss with an adversarial loss for distinguishing samples from $\mathbf{h}^I$ and samples from $\mathbf{h}^C$. By minimizing the adversarial loss, we aim to require $\mathbf{h}^I$ and $\mathbf{h}^C$ to have similar distributions and contain shared information from different user aspects (reflecting user-item and user-curator preferences). Then we will be able to maximize the effectiveness of $\mathbf{h}$ as the shared latent representation for both tasks.

Thus we have a discriminator designed to predict whether a vector is encoded from the curator feedback vector ($\mathbf{h}^C$) or from the item feedback vector ($\mathbf{h}^I$) (see Figure 3). The discriminator can be a simple neural network consisting of several fully-connected layers. This binary classifier is trained to classify samples from $\mathbf{h}^C$ as positive (1) and samples from $\mathbf{h}^I$ as negative (0). We use binary cross-entropy to train the discriminator.

While training, we follow the *minimax* strategy. That is, in each batch, we will first try to optimize the discriminator in classifying latent representations from different aspects (*Max*). And then we minimize the reconstruction loss while also misleading the discriminator (*Min*). Let θ_D, e and l denote the parameters, predicted labels and ground truth labels for the discriminator. In the *Max* step, the objective function is to maximize the adversarial loss L_{adv}:

$$L_{adv} = \max_{\theta_D} \frac{1}{b} \sum_{b} ((1-l)\ln(1-e) + l \ln e)) \tag{3}$$

where b is the number of input samples for the discriminator. And then in the *Min* step, the encoders need to generate latent representations to mislead the discriminator. Thus we supplement the reconstruction loss L_{rec} in Equation 2 with the adversarial loss L_{adv} and the total loss of the combined model becomes:

$$L = \min_{\theta_R} \max_{\theta_D} L_{rec} + \lambda L_{adv} \tag{4}$$

in which θ_R denotes the whole set of parameters for the model excluding θ_D. And λ is the hyper-parameter used to adjust the weight for the adversarial loss.

3.4 RQ3: Personalized Fusing via Attention

So far, by doing a naive addition, we assign the same weight on $\mathbf{h}^C$ and $\mathbf{h}^I$ to generate the shared latent representation $\mathbf{h}$, which may not reflect the personal preferences of users. For different users, the correlations between their preferences on curators and preferences on items could be different. Some users may prefer to follow curators who share a similar preference for items with them (e.g., a user who likes science fiction novels follows curators who also focus on science fiction). Other users may prefer to follow curators with novel content to gain a wider exposure (e.g., a user who mostly reads science fiction novels may follow a curator who prefers biographies). Thus, we need to perform a weighted sum on $\mathbf{h}_u^C$ and $\mathbf{h}_u^I$, where the weight on each component denotes its influence in both tasks for user u. We replace the simple addition between $\mathbf{h}^C$ and $\mathbf{h}^I$ with an attention layer [4, 7, 42] to produce user-specific (personalized) weighted aggregations of latent representations.

As explained in Figure 3, For user u, we will first pass the one-hot representation $\mathbf{u}$ to a fully-connected embedding layer and generate a unique latent representation $\mathbf{E}_u = \mathbf{W}^E\mathbf{u} + \mathbf{b}^E$, in which $\mathbf{W}_E$ is the weight matrix and $\mathbf{b}^E$ is the bias. The user embedding will be used as a user-aware coefficient to adjust the importance of latent representations from different components in the attention layer. Thus for user u, the weights α_u^C and α_u^I for latent representations $\mathbf{h}_u^C$ and $\mathbf{h}_u^I$ are calculated as:

$$\alpha_u^C = \frac{\exp O_u^C}{\exp O_u^C + \exp O_u^I} \qquad \alpha_u^I = \frac{\exp O_u^I}{\exp O_u^C + \exp O_u^I}$$

where

$$O_u^S = \mathbf{z}^T \cdot \tanh(\mathbf{P}\mathbf{h}_u^S + \mathbf{Q}\mathbf{E}_u + \mathbf{b}^A), \quad S \in \{C, I\}$$

in which $\mathbf{P}$ and $\mathbf{Q}$ are the weight matrices of the attention layer. $\mathbf{z}$ is a transform vector and $\mathbf{b}^A$ represents the bias. Then for user u, the latent representation $\mathbf{h}_u^C$ and $\mathbf{h}_u^I$ will be aggregated based on the weights α_u^C and α_u^I correspondingly. Empirically we find that this user-specific weighted sum can achieve better results than the simple addition for both tasks.

Isolated Hidden Layer for Primary Task. We have reached a personalized-aggregated latent representation $\mathbf{h}$ containing shared information for both primary and supplementary tasks. However, there is task-specific information which can assist one task but become noise for another task [23]. Thus in order to further enhance the performance in our primary task of recommending *curators*, we reserve latent factors for information targeting the primary task only. Thus, we add an isolated hidden layer $\overline{\mathbf{h}}$ accompanying with the shared layer $\mathbf{h}$ for our primary task. $\overline{\mathbf{h}}$ can be generated with Equation 1 but with a separate weight matrix and bias ($\overline{\mathbf{V}}$ and $\overline{\mathbf{b}}$). It is concatenated with the shared latent $\mathbf{h}$ before decoding for the primary task. Then we get our final model as shown in Figure 3.

4 EXPERIMENTS

In this section, we evaluate the proposed CuRe model via experiments over two real-world curation datasets. We seek to answer the following questions: (i) How does the proposed framework perform in the primary task of recommending curators? (ii) What is the impact of the different components in CuRe on the quality of curator recommendation? (iii) While CuRe aims to improve curator recommendation (the primary task), how well does it perform on the supplementary task of item recommendation? (iv) How does CuRe perform under the cold-start scenario?

4.1 Datasets

To examine how the proposed framework performs in real-world scenarios, we conduct our experiments on datasets (summarized in Table 1) from two curation platforms supporting different curating behaviors. In Spotify, users curate music into playlists. Goodreads enable users to not only curate books into bookshelves but also leave feedback including ratings, reviews, and so on [36]. We split each dataset and use 60% for training, while using 10% for validation and 30% for testing.

Goodreads. Since user IDs in Goodreads are consecutive integer numbers, we randomly select 1 million users and crawl the lists of users they follow. We keep active users with more than 5 followers or 5 followees. For each user, we crawl the list of books they have left feedback on (putting to bookshelves, ratings, reviews or tags). Further, we consider all books with more than 50 feedback, resulting in more than 60,000 books listed by about 50,000 users.

Dataset	User	Item	User-User Interactions	User-Item Interactions
Goodreads	48,208	61,848	528,816	10,526,215
Spotify	25,471	70,107	227,024	4,499,741

Table 1: Summary of Datasets

Spotify. Spotify is one of the largest music streaming platforms, on which users can create and share their playlists. Users can follow other users or playlists and then get updates from them. To sample from Spotify, we create a seed list of playlists by issuing 200 keyword queries representing popular topics on Spotify (e.g., pop, coffee, trip) and then randomly selecting 0.2 million returned playlists. We then identify the users who curate these playlists and crawl their followees, arriving at a list of 5 million valid user IDs. We then identify active users with more than 5 followers or 5 followees. For each user, we consider all the music tracks in playlists they curate as what they have left feedback on. We filtered out all tracks which have received less than 50 feedback.

4.2 Setup

Metrics. To evaluate the quality of top-k recommendation, we adopt the F1 score and normalized discounted cumulative gain (NDCG). F1 is the combination of recall and precision, where recall is the percentage of relevant curators discovered by the top-k recommendations, while precision represents the fraction of correctly predicted curators in the top-k recommendations. Let F1@K, P@K and R@K represent the F1, precision and recall while considering the top-k ranked list from each model. Then the F1 score is calculated as $F1@K = \frac{2*P@K*R@K}{P@K+R@K}$.

Taking the positions of recommendations into consideration, we also measure NDCG, which is the ratio between discounted cumulative gain (DCG) and ideal discounted cumulative Gain (IDCG). Here $DCG@K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)}$, $IDCG@K = \sum_{i=1}^{min(|REL|,K)} \frac{1}{\log_2(i+1)}$, in which rel_i denotes the relevance score the recommendation with rank i. If the recommendation is in the test set, then $rel_i = 1$, otherwise, $rel_i = 0$. $|REL|$ represents the size of the test set. Then NDCG is calculated as $NDCG@K = \frac{DCG@K}{IDCG@K}$.

Baselines. We consider a suite of baselines from simple models to recently introduced state-of-the art models, which can be applied to generate user recommendation[1]:

- *Most Popular (MP).* This model ranks curators (and items) based on their popularity and recommends the most popular ones.
- *User-based Collaborative Filtering (UCF).* The similarity between users is defined by the cosine similarity of their feedback vectors. The recommendation for a user is generated by the weighted aggregation of similar users' feedback vectors.

[1]In our preliminary experiments, we also tested graph-based link prediction (like DeepWalk, node2vec [13]). We find that they perform much worse than the basic collaborative filtering (UCF), which indicates that graph-based methods are not well-suited for this curator recommendation task.

	Goodreads					Spotify				
	F1		NDCG		Ave Δ	F1		NDCG		Ave Δ
	K=5	K=10	K=5	K=10		K=5	K=10	K=5	K=10	
MP	0.0559	0.0641	0.0678	0.0884	276.87%	0.0957	0.0809	0.1224	0.1316	40.33%
UCF	0.1536	0.1455	0.1925	0.2181	45.89%	0.1028	0.0936	0.1313	0.1472	26.93%
Implicit	0.1744	0.1680	0.2151	0.2466	28.62%	0.1029	0.0964	0.1229	0.1420	29.28%
BPR	0.1700	0.1653	0.2122	0.2416	31.09%	0.1078	0.0975	0.1363	0.1525	21.91%
EMJ	0.1736	0.1729	0.2129	0.2485	27.99%	0.1069	0.1016	0.1283	0.1495	23.44%
AMF	0.1807	0.1739	0.2251	0.2552	23.89%	0.1127	0.1036	0.1417	0.1599	16.22%
DAE	0.1893	0.1819	0.2353	0.2668	18.43%	0.1240	0.1124	0.1560	0.1749	6.14%
VAE	0.1910	0.1819	0.2386	0.2708	17.31%	0.1223	0.1099	0.1530	0.1715	8.16%
CDAE	0.1894	0.1822	0.2357	0.2673	18.26%	0.1266	0.1131	0.1583	0.1761	4.86%
Joint-DAE	0.2142	0.1998	0.2693	0.3022	5.10%	0.1259	0.1138	0.1577	0.1768	4.84%
CuRe	**0.2252**∗	**0.2089**∗	**0.2840**∗	**0.3181**∗	-	**0.1317**∗	**0.1191**∗	**0.1656**∗	**0.1858**∗	-

Table 2: Comparing Models on Top-K *Curator* Recommendation for K=5 and K=10. ∗ indicates that the improvement of the best result is statistically significant compared with other methods for $p < 0.01$.

- *Bayesian Personalized Ranking with Matrix Factorization (BPR).* This model [31] conducts matrix factorization with pairwise ranking from implicit feedback.
- *Implicit Matrix Factorization (Implicit).* This model [17] extends the basic matrix factorization (MF) to fit the implicit recommendation scenarios. It assumes that users' preference can be inferred from the frequency of interaction.
- *Adversarial Matrix Factorization (AMF).* This model [15] utilizes adversarial training to enhance the robustness of latent factors in the matrix factorization component of BPR.
- *Denoising Autoencoder (DAE).* DAE is a generalization of collaborative filtering and explained in detail in Section 3.2.
- *Collaborative Denoising Autoencoder (CDAE).* This model [41] extends DAE by adding a user node with each feedback vector input to form a more flexible structure.
- *Variational Autoencoder for Collaborative Filtering (VAE).* This model [22] extends DAE by adopting multinomial likelihood and using Bayesian inference for parameter estimation.

We also consider two additional approaches that can be applied for user recommendation by considering both user-user interactions and user-item interactions:

- *Embedding Factorization Models for Joint Recommendation (EMJ).* This model [5] can jointly recommend curators and items by combining the factorization model for user-item interactions and the factorization model for user-user interactions.
- *Joint Curator-Item-DAE (Joint-DAE).* This is a simplified version of CuRe without the adversarial learning process and the attention layer. This model (in Section 3.2) enhances basic DAE with a supplementary task to recover the feedback on items.

Parameter Settings. All the experiments were conducted on a server machine with Intel i7 CPU and a 12 GB Nvidia GeForce Titan XP GPU. For BPR and AMF, we use the implementation provided by [15]. We use the implementation in [5] for EMJ. We do a grid search for latent factor size over {50, 100, 150, 200, 300, 400, 500} and follow the original settings in the code for other parameters.

For all the models containing an autoencoder structure, we implement them in Keras. We do a grid search for best latent factors size over {50, 100, 150, 200, 300, 400, 500} for each model. We report their best results in Table 2 and 3. While training the model, we use the L2-norm as regularization for each layer. We use a dropout layer to add the mask-out noise for each DAE. The dropout rate is set to be 0.3 for all the models. The batch size is set to be 256. Negative sampling rate is set to be 10 for all the methods. That is for each observed feedback, we randomly select 10 unobserved points as negative samples. We fine-tune α and λ to balance the tasks. While building up models utilizing both feedback on curators and items, instead of randomly initializing the weight matrices and bias for encoders, we load the parameters from the pre-trained DAE models. In CuRe, the discriminator is a simple neural network with one hidden layer. And the size of the hidden layer is half the size of $\mathbf{h}^C$ and $\mathbf{h}^I$. For all the models, we run each of them 10 times and report the average NDCG and F1.

4.3 Curator Recommendation

To evaluate the overall performance of the proposed CuRe model in curator recommendation, we compare it with several baseline models by F1@K and NDCG@K for K=5 and K=10 (see Table 2). On both Goodreads and Spotify, CuRe achieves the best results, demonstrating its effectiveness for curator top-K recommendation. In Goodreads, collaborative filtering (UCF) can improve the naive method of recommending the most popular (MP) curators by 158.10%. However, UCF just improves upon MP by 10.56% in Spotify. In Goodreads, there are many methods for users to discover others who share similar interest in books. For example, users can read reviews and also check users who write them. Goodreads also support reading groups in which users can share information in a certain area. Thus similar users tends to follow similar curators in Goodreads. In contrast, the Spotify App mainly recommends popular curators. Alternatively, users can discover new curators by searching for playlists they are interested in. Thus, methods with collaborative filtering (UCF) or matrix factorization (Implicit) have difficulty improving on MP as much as in Goodreads.

We can see that the pairwise personalized ranking method – BPR performs similarly to Implicit in Goodreads while outperforming Implicit by 6.05% in Spotify. Using adversarial training for pairwise learning, AMF can enhance the robustness of BPR and thus lead

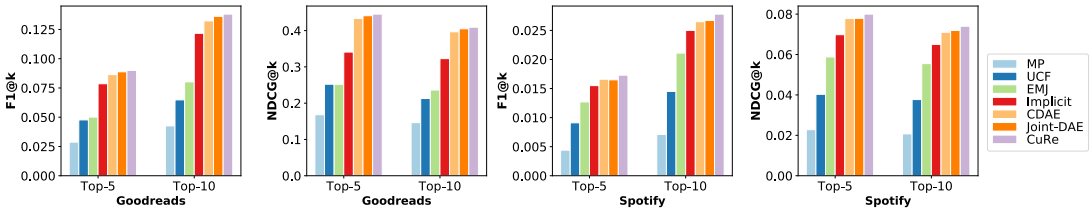

Figure 4: Top-k Item Recommendation Performance (corresponding to results in Table 2). The improvement of CuRe is statistically significant compared with all other methods for $p < 0.05$ on F1 and NDCG.

to a 5.80% (Goodreads) and 4.90% (Spotify) improvement. But we can see that denoising autoencoder (DAE), which we pick as the basic structure for preference elicitation, performs even better than AMF, demonstrating its superiority for this curator recommendation problem with sparse feedback.

EMJ utilizes both user-curator feedback and user-item feedback while generating embedding for users (curators) and items. In Goodreads, with the supplementary information and structure to extract the interaction between curators and items, EMJ can outperform BPR by 2.48%, which shows the importance of incorporating user preferences on individual items to improve curator recommendation. Further, we can see that Joint-DAE can improve DAE by 12.68% by utilizing the supplementary item feedback. This improvement shows the effectiveness of the the proposed structure in combining both types of information. However, because user feedback on items is sparser in Spotify, we find that the improvement from DAE to Joint-DAE is smaller. In both Goodreads and Spotify, CuRe can improve Joint-DAE by about 5%. We can see that the adversarial learning and attention layer are beneficial for top-k curator recommendation problem in which the feedback information is sparse and noisy.

4.4 Item Recommendation

In our curator recommendation, we can also infer users' preference on items with the supplementary task. We can take advantage of the autoencoder structure in this model to recommend a ranked list of items for each user. Thus we compare it with some baselines under top-K recommendation for items (Figure 4). For EMJ, Joint-DAE and CuRe which can recommend both curators and items at the same time, we list the NDCG and F1 they achieve for item recommendation corresponding to the results in Table 2. We can see that CuRe performs better then EMJ for both tasks. And while recommending items, feedback on curators can be treated as context information. CuRe can even outperform the CDAE model, which is more powerful than basic DAE, in item recommendation. Further, the improvement in Goodreads is smaller than that in Spotify. It is possible that users in Spotify are more likely to collect music tracks from playlists created by curators they follow while creating their own playlists. So by who they follow or who they may follow, we can infer the items they are interested in.

4.5 Evaluating CuRe Design

The proposed CuRe model extends the basic DAE with several components. To evaluate the effectiveness of each component, we start from DAE and add supplementary task, discriminator, isolated layer, and the attention layer sequentially. We show their performance (F1@10) for both datasets in Figure 5:

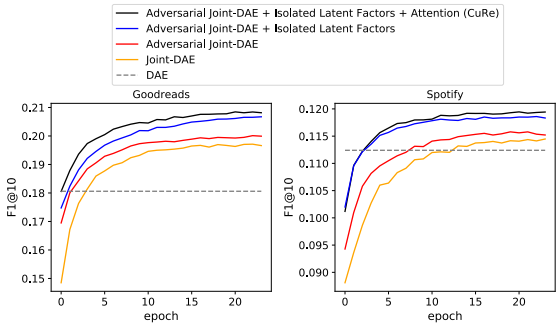

Figure 5: Comparison between different models on F1@10.

Supplementary Task and Information. The dotted lines denote the best results we can achieve with the basic DAE. By introducing the item feedback with the Joint Curator-Item-DAE (Joint-DAE) model, there is a 9.14% (Goodreads) and 1.25% (Spotify) improvement on F1@10, which demonstrates that users' preferences on items can assist in predicting their preference on curators. Utilizing the item feedback information can help to compensate the sparsity of feedback on curators. Further, we find that the model brings in a much larger improvement in Goodreads because there is denser user-item interaction information in Goodreads.

Adversarial Learning and Discriminator. After adding the discriminator to *Joint-DAE*, we see that the resulted Adversarial Joint-DAE can further improve *Joint-DAE* in Goodreads and Spotify (from orange line to the red line in Figure 5). Since adversarial loss and the discriminator can guide the hidden latent representation for the primary task and supplementary task to be similar to each other (that is, each should share a similar distribution), we find the Adversarial Joint-DAE can achieve better performance while training for the same number of epochs and thus converge in fewer epochs.

Isolated Layer. In CuRe, besides $\mathbf{h}$ containing shared information for both tasks, we also design an isolated layer $\overline{\mathbf{h}}$ to encode information which only benefit the primary task. After appending the isolated hidden layer, we can see F1@10 can be improved. Thus it is necessary for us to reserve some isolated latent factors to represent users' preference on curators. For example, there are users who follow curators based on popularity but not based on items the curators collect. We should have an isolated layer for information which can support the primary task but can add noise for the supplementary task. It is able to improve the curator recommendation with higher accuracy and more robustness.

Attention Layer. Recently, neural attention mechanism has been widely applied on neural models for different applications. In CuRe, while integrating the preferences on curators and preferences on items, we replace the addition with an attention layer to conduct

	Goodreads					Spotify				
	F1		NDCG		Ave Δ	F1		NDCG		Ave Δ
	K=5	K=10	K=5	K=10		K=5	K=10	K=5	K=10	
MP	0.0554	0.0556	0.0755	0.1000	243.36%	0.0815	0.0644	0.1084	0.1223	44.51%
UCF	0.1401	0.1167	0.1530	0.1283	82.84%	0.0892	0.0759	0.1192	0.1399	27.99%
Implicit	0.1530	0.1283	0.2079	0.2479	33.14%	0.0881	0.0769	0.1116	0.1338	31.59%
BPR	0.1393	0.1187	0.1847	0.2231	47.04%	0.0893	0.0776	0.1156	0.1378	28.73%
EMJ	0.1562	0.1358	0.2067	0.2526	30.26%	0.0912	0.0835	0.1132	0.1400	26.02%
AMF	0.1469	0.1245	0.1952	0.2349	39.58%	0.0914	0.0801	0.1195	0.1429	24.79%
DAE	0.1607	0.1337	0.2176	0.2583	27.38%	0.1047	0.0903	0.1368	0.1623	9.61%
VAE	0.1681	0.1378	0.2293	0.2697	22.04%	0.1056	0.0888	0.1379	0.1613	9.78%
CDAE	0.1608	0.1352	0.2180	0.2605	26.68%	0.1079	0.0903	0.1404	0.1639	7.81%
Join-DAE	0.1889	0.1531	0.2607	0.3044	8.46%	0.1045	0.0909	0.1374	0.1636	9.16%
CuRe	**0.2040**∗	**0.1652**∗	**0.2844**∗	**0.3314**∗	-	**0.1160**∗	**0.0974**∗	**0.1514**∗	**0.1771**∗	-

Table 3: Comparing Models on Top-K *Curator* Recommendation under *Cold-start Setting*. ∗ indicates that the improvement of the best result is statistically significant compared with other methods for $p < 0.01$.

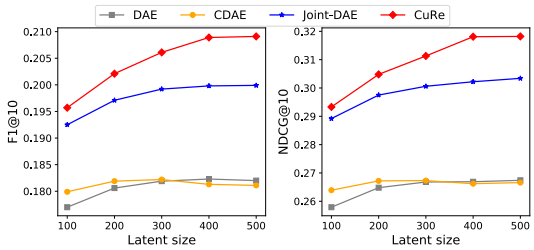

Figure 6: Curator Recommendation vs. Latent Factors.

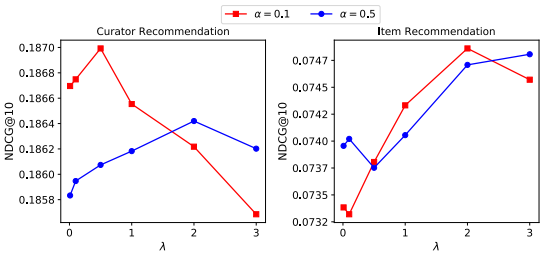

Figure 7: Curator recommendation (primary task, left) vs. item recommendation (supplemental task, right).

a user-specific weighted sum. In Figure 5, we can see that there is gap between the blue line and the black line, which indicates the improvement of our curator recommendation by adopting the attention mechanism. And further, with the user-dependent weights, we can improve the performance of the supplementary task for item recommendation while also improving curator recommendation.

4.6 Hyper-Parameters

Latent Factors. To examine the influence of the hidden layer (latent factors), we compare the performance of CuRe with baselines which contain an autoencoder structure in Goodreads (see Figure 6) by varying the size of the hidden layer. For fair comparison, we use the sum of the size for the isolated layer and the shared layer as the size of the latent factors for CuRe. When the size of latent factors is relatively small, as the additional user node can help to store part of the information, CDAE performs better than DAE. But when the size of latent factors goes up to 300, DAE achieves similar performance as CDAE since the basic DAE structure is enough for information from user feedback on curators. Further, we can see that Joint-DAE can outperform both CDAE and DAE even with a smaller size of latent factors. Thus we can see the necessity of incorporating the supplementary information of user feedback on items for curator recommendation. We use an isolated layer in CuRe for information for the primary task only. Comparing CuRe with Joint-DAE, we can see while the size of latent factor is small, CuRe improve Joint-DAE slightly as small-sized isolated layer and shared layer can weaken the power of the model. However, when the latent factor is ample for capturing user preferences extracted from feedback for both items and curators, CuRe performs much better than Joint-DAE, which demonstrates the effectiveness of the proposed model design. We see a similar pattern in Spotify.

Primary Task vs. Supplementary Task. In CuRe, we use α to adjust the weight of the supplementary task, and λ to balance the adversarial loss while fusing latent representations for both tasks (in Equation 4). To get insight into how they impact the performance of both tasks, we do a case study to compare the performance of CuRe under various λ and α in Spotify. We plot the resulting NDCG@10 under the situation with a *light* weight ($\alpha = 0.1$) or a *heavy* weight ($\alpha = 0.5$) on the supplementary task (in Figure 7). Appropriate amount of adversarial loss can benefit both the curator recommendation and item recommendation while allocating different weights on the supplementary task, which shows the benefit of the adversarial process in fusing the latent factors. Without the adversarial loss, while putting a *light* weight on the supplementary task, we can see that the performance on item recommendation is much worse than the situation that we give it a *heavy* weight. But *heavy* weight on the supplementary task can be harmful for the primary task. By introducing the adversarial loss, we can see that the performance on both tasks increases. And the model with *light* weight on supplementary task is more sensitive to the adversarial loss. With *light* weight, we can see that if we set $\lambda \in (0.5, 1.0)$, both tasks can achieve better results comparing to the situation with either *light* or *heavy* weight on supplementary task but no adversarial loss ($\lambda = 0$). This case study illustrates how *the adversarial learning can help to compensate for the loss in supplementary task (item recommendation) while improving curator recommendation.*

4.7 Cold Start

Recommender systems commonly come across cold start problem in which new users have not left enough feedback for revealing their preferences. There are users in Goodreads or Spotify who just follow a few curators and we still want to make accurate predictions as to who they are going to follow. Thus we compare CuRe with other methods under a cold start setting in Table 3. We test each model on the 30% of users who left the least numbers of feedback on curators. We can see CuRe improves Joint-DAE, which is the baseline with best performance, by 8.46% and 9.61% in Goodreads and Spotify correspondingly. The improvement of CuRe compared with other models under the cold-start setting is larger than that in the normal setting. Thus we can see that the proposed model can compensate for sparsity and provide robust recommendation.

5 CONCLUSION

We have proposed a new user recommendation model in content curation platforms enhanced with adversarial learning and an attentive mechanism to fuse users preferences on curators and preferences on items. Through experiments on data sampled from both Goodreads and Spotify, we find that the proposed model can outperform state-of-the-art baselines in F1 and NDCG in curator recommendation. Meanwhile, CuRe also demonstrates good performance on the supplementary task of item recommendation. In the future, we are interested in further exploring the interactions between users and curators. To get more insight of the great human power hidden in curation communities, we would like to capture the dynamics of users and curators along time via different neural models.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant IIS-1841138.

REFERENCES

[1] Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *ACL*.
[2] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *RecSys*.
[3] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In *MM*.
[4] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive group recommendation. In *SIGIR*.
[5] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*.
[6] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.
[7] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*.
[8] Zhiyuan Cheng, James Caverlee, Himanshu Barthwal, and Vandana Bachani. 2014. Who is the barbecue king of texas?: a geo-spatial approach to finding local experts on twitter. In *SIGIR*.
[9] Zhiyong Cheng, Jialie Shen, Lei Zhu, Mohan S Kankanhalli, and Liqiang Nie. 2017. Exploiting Music Play Sequence for Music Recommendation.. In *IJCAI*.
[10] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, Nitesh V Chawla, Jinghai Rao, and Huanhuan Cao. 2012. Link prediction and recommendation across heterogeneous social networks. In *ICDM*. IEEE.
[11] Kevin Duh, Tsutomu Hirao, Akisato Kimura, Katsuhiko Ishiguro, Tomoharu Iwata, and Ching-Man Au Yeung. 2012. Creating Stories: Social Curation of Twitter Messages.. In *ICWSM*.
[12] Hancheng Ge, James Caverlee, and Haokai Lu. 2016. Taper: A contextual tensor-based approach for personalized expert recommendation. In *RecSys*.
[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.
[14] John Hannon, Mike Bennett, and Barry Smyth. 2010. Recommending twitter users to follow using content and collaborative filtering approaches. In *RecSys*.
[15] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*.
[16] Yun He, Jianling Wang, Wei Niu, and James Caverlee. 2019. A Hierarchical Self-Attentive Model for Recommending User-Generated Item Lists. In *CIKM*.
[17] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*.
[18] Krishna Yeswanth Kamath, Ana-Maria Popescu, and James Caverlee. 2013. Board Recommendation in Pinterest.. In *UMAP Workshops*.
[19] Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2018. Recommendation Through Mixtures of Heterogeneous Item Relationships. In *CIKM*.
[20] Giannis Karamanolakis, Kevin Raji Cherian, Ananth Ravi Narayan, Jie Yuan, Da Tang, and Tony Jebara. 2018. Item Recommendation with Variational Autoencoders and Heterogeneous Priors. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*. ACM, 10–14.
[21] Koki Kitaya, Hung-Hsuan Huang, and Kyoji Kawagoe. 2012. Music curator recommendations using linked data. In *Innovative Computing Technology (INTECH), 2012 Second International Conference on*. IEEE, 337–339.
[22] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*.
[23] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *ACL*.
[24] Yuchen Liu, Dmitry Chechik, and Junghoo Cho. 2016. Power of human curation in recommendation system. In *WWW (Companion Volume)*.
[25] Yidan Liu, Min Xie, and Laks VS Lakshmanan. 2014. Recommending user generated item lists. In *RecSys*.
[26] Haokai Lu and James Caverlee. 2015. Exploiting geo-spatial preference for personalized expert recommendation. In *RecSys*.
[27] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: multi-task learning for recommendation and explanation. In *RecSys*.
[28] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).
[29] David W McDonald and Mark S Ackerman. 2000. Expertise recommender: a flexible recommendation system and architecture. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 231–240.
[30] Marco Pennacchiotti and Siva Gurumurthy. 2011. Investigating topic models for social media user recommendation. In *WWW (Companion Volume)*.
[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.
[32] Daniel Schall. 2014. Who to follow recommendation in large-scale online development communities. *Information and Software Technology* 56, 12, 1543–1555.
[33] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*.
[34] Kai Shu, Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2018. Crossfire: Cross media joint friend and item recommendations. In *WSDM*.
[35] Jianshan Sun, Gang Wang, Xusen Cheng, and Yelin Fu. 2015. Mining affective text to improve social media item recommendation. *Information Processing & Management* 51, 4 (2015), 444–457.
[36] Mike Thelwall and Kayvan Kousha. 2017. Goodreads: A social network site for book readers. *Journal of the Association for Information Science and Technology* 68, 4 (2017), 972–983.
[37] Reggie Ugwu. 2016. Inside The Playlist Factory. https://www.buzzfeed.com/reggieugwu/the-unsung-heroes-of-the-music-streaming-boom
[38] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *RecSys*.
[39] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, Dec (2010), 3371–3408.
[40] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. *SIGIR* (2018).
[41] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *CIKM*.
[42] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).
[43] Zhenlei Yan and Jie Zhou. 2012. User recommendation with tensor factorization in social networks. In *ICASSP*.
[44] Xitong Yang, Yuncheng Li, and Jiebo Luo. 2015. Pinterest board recommendation for twitter users. In *MM*.
[45] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. 2015. Improving user topic interest profiles by behavior factorization. In *WWW*.
[46] Tom Chao Zhou, Hao Ma, Michael R Lyu, and Irwin King. 2010. UserRec: A User Recommendation Framework in Social Tagging Systems.. In *AAAI*.