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.

CCLS CONCEPTS

- Information systems → Recommender systems;

KEYWORDS

Recommendation; Content Curation; Attentive Adversarial Model

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. 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
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 Curation platform (CuRe) user Recommendation. 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 $U = \{u_1, u_2, ..., u_N\}$ 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 $U$. 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 $y_u = [y_{u1}, y_{u2}, ..., y_{uN}]$ 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 $I = [i_1, i_2, ..., i_P]$ denote the set of items we consider. Similarly, we use binary vector $t_u$ 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, $y_u \in U$, we want to recommend a ranked list of curators to follow $C_u \subseteq U$ (which we refer to as our primary task) and a ranked list of interested items $I_u \subseteq I$ (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 $y_u = [y_{u1}, y_{u2}, ..., y_{uN}]$ 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 $y_u$ with a probability of $\eta$ and then we get the corrupted vector $\tilde{y}_u = [\tilde{y}_{u1}, \tilde{y}_{u2}, ..., \tilde{y}_{uN}]$. We feed this vector into the encoder and generate the hidden latent representation:

$$h_u = \delta(V\tilde{y}_u + b),$$  \hspace{1cm} (1)

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

$$\tilde{y}_u = \delta(Wh_u + b'),$$

where $W$ and $b'$ 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.

$$L = \sum_{u \epsilon U} \left( \sum_{c \epsilon S_u^C} (\tilde{y}_{uc} - y_{uc})^2 \right)$$

Practically, while considering the construction loss, besides the users with positive feedback from user $u$, $S_u^C$ 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 $\tilde{y}_u$, in which $\tilde{y}_{uc}$ represents $u$’s preference on $c$. Such an approach, however, builds a representation $h_u$ 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 $h_u$ 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 $h_u^C$ denote the hidden latent factors for user $u$ generated from this supplementary DAE. We supplement $h_u^C$ with the user-specific vector $h_u^I$ 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 $\tilde{y}_u$ embedded from user-curator interactions (with Equation 1) as $h_u^C$. Given the supplementary item feedback vector $t_u$ of user $u$, we first corrupt them with the mask-out noise (denoising). And then encode them into hidden representations $h_u^I$:

$$h_u^I = \delta(V^I t_u + b')$$

where $t_u$ is the corrupted version of $t_u$. Then we construct the enriched version of $h_u$ by integrating the hidden representations $h_u^I$ and $h_u^C$ via element-wise addition $h_u = h_u^C + h_u^I$.

We feed the combined hidden representation $h_u$ of user $u$ into separate decoders for the primary task and supplementary task.

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

$$L_{rec} = \sum_{u \epsilon U} \left( \sum_{c \epsilon S_u^C} (\tilde{y}_{uc} - y_{uc})^2 + \alpha \sum_{i \epsilon S_u^I} (\tilde{y}_{ui} - y_{ui})^2 \right),$$

where $S_u^I$ 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 $\alpha$ to adjust the weight for the supplementary task.
3.3 RQ2: Fusing Latent Representations

In our approach so far, the combined representation $h$ – which is the element-wise aggregation of samples in $h^I$ and samples in $h^C$ – 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 $h^I$ and $h^C$ should correspond to the same latent factor of users while forming the shared representation. That is, we would like samples in $h^I$ and $h^C$ to have similar distributions and 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 $h^I$ and samples from $h^C$. By minimizing the adversarial loss, we aim to require $h^I$ and $h^C$ 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 $h$ as the shared latent representation for both tasks.

Thus we have a classifier designed to predict whether a latent representation is from $h^I$ or $h^C$. The attention layer can generate user-specific weighted aggregations $h$ for user’s preferences on curators and items.

![Figure 3: The structure of CuRe. The model generates user's latent presentations with feedback vectors on curators and items.](image)

As in the work of [23], we supplement the reconstruction loss with an adversarial loss for distinguishing samples from $h^I$ and samples from $h^C$. By minimizing the adversarial loss, we aim to require $h^I$ and $h^C$ 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 $h$ as the shared latent representation for both tasks.

Mathematically, we define the adversarial loss as follows:

$$L_{adv} = \max_{\theta_D} \frac{1}{b} \sum_{i=1}^{b}[(1-l) \ln(1-e) + l \ln e]$$  \hspace{1cm} (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:

$$L = \min_{\theta_B} \max_{\theta_D} L_{rec} + \lambda L_{adv}$$  \hspace{1cm} (4)

in which $\theta_B$ denotes the whole set of parameters for the model excluding $\theta_D$, and $\lambda$ 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 $h^C$ and $h^I$ to generate the shared latent representation $h$, 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 $h^C$ and $h^I$ 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 $u$ to a fully-connected embedding layer and generate a unique latent representation $E_u = W_E u + b_E$, in which $W_E$ is the weight matrix and $b_E$ 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 $a_u^C$ and $a_u^I$ for latent representations $h_u^C$ and $h_u^I$ are calculated as:

$$a_u^C = \frac{\exp O_u^C}{\exp O_u^C + \exp O_u^I} \quad a_u^I = \frac{\exp O_u^I}{\exp O_u^C + \exp O_u^I}$$

where $O_u^S = z^T \cdot \tanh(P h_u^S + Q E_u + b^A)$, $S \in \{C, I\}$

in which $P$ and $Q$ are the weight matrices of the attention layer. Then for user $u$, the latent representation $h_u^C$ and $h_u^I$ will be aggregated based on the weights $a_u^C$ and $a_u^I$ 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 $h$ 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 $\bar{h}$ accompanying with the shared layer $h$ for our primary task. $\bar{h}$ can be generated with Equation 1 but with a separate weight matrix and bias ($V$ and $\bar{b}$). It is concatenated with the shared latent $h$ 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.

**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 $F1@K = \frac{2 \cdot P@K \cdot R@K}{P@K + R@K}$.

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 $\text{DCG@K} = \sum_{i=1}^{K} rel_i \frac{\text{DCG}_i(K)}{\log(1+i)}$, $\text{IDCG@K} = \sum_{i=1}^{K} \text{top_i(K)} \frac{1}{\log(1+i)}$, 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, $rel_i = 0$. $\text{top_i(K)}$ represents the size of the test set. Then NDCG is calculated as $\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}$.

#### 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.

2. **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.

3. **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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User</th>
<th>Item</th>
<th>User-User Interactions</th>
<th>User-Item Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodreads</td>
<td>48,208</td>
<td>61,848</td>
<td>528,816</td>
<td>10,526,215</td>
</tr>
<tr>
<td>Spotify</td>
<td>25,471</td>
<td>70,107</td>
<td>227,024</td>
<td>4,499,741</td>
</tr>
</tbody>
</table>

Table 1: Summary of Datasets
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 hC and hI. For all the models, we run each of them 10 times and report the average NDCG and F1.

<table>
<thead>
<tr>
<th></th>
<th>Goodreads F1</th>
<th>NDCG</th>
<th>Ave Δ</th>
<th>Spotify F1</th>
<th>NDCG</th>
<th>Ave Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K=5</td>
<td>K=10</td>
<td></td>
<td>K=5</td>
<td>K=10</td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td>0.0559</td>
<td>0.0641</td>
<td>0.0678</td>
<td>0.0884</td>
<td>276.87%</td>
<td>0.0957</td>
</tr>
<tr>
<td>UCF</td>
<td>0.1536</td>
<td>0.1455</td>
<td>0.1925</td>
<td>0.2181</td>
<td>45.89%</td>
<td>0.1028</td>
</tr>
<tr>
<td>Implicit</td>
<td>0.1744</td>
<td>0.1680</td>
<td>0.2151</td>
<td>0.2466</td>
<td>28.62%</td>
<td>0.1029</td>
</tr>
<tr>
<td>BPR</td>
<td>0.1700</td>
<td>0.1653</td>
<td>0.2122</td>
<td>0.2416</td>
<td>31.09%</td>
<td>0.1078</td>
</tr>
<tr>
<td>EMJ</td>
<td>0.1736</td>
<td>0.1729</td>
<td>0.2129</td>
<td>0.2485</td>
<td>27.99%</td>
<td>0.1069</td>
</tr>
<tr>
<td>AMF</td>
<td>0.1807</td>
<td>0.1739</td>
<td>0.2251</td>
<td>0.2552</td>
<td>23.89%</td>
<td>0.1127</td>
</tr>
<tr>
<td>DAE</td>
<td>0.1893</td>
<td>0.1819</td>
<td>0.2353</td>
<td>0.2668</td>
<td>18.43%</td>
<td>0.1240</td>
</tr>
<tr>
<td>VAE</td>
<td>0.1910</td>
<td>0.1819</td>
<td>0.2386</td>
<td>0.2708</td>
<td>17.31%</td>
<td>0.1223</td>
</tr>
<tr>
<td>CDAE</td>
<td>0.1894</td>
<td>0.1822</td>
<td>0.2357</td>
<td>0.2673</td>
<td>18.26%</td>
<td>0.1266</td>
</tr>
<tr>
<td>Joint-DAE</td>
<td>0.2142</td>
<td>0.1998</td>
<td>0.2693</td>
<td>0.3022</td>
<td>5.10%</td>
<td>0.1259</td>
</tr>
<tr>
<td>CuRe</td>
<td>0.2252+</td>
<td>0.2089+</td>
<td>0.2840+</td>
<td>0.3181+</td>
<td>-</td>
<td>0.1317*</td>
</tr>
</tbody>
</table>

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.

## 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 contrast, the Spotify App mainly recommends in Goodreads. In Goodreads, 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
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:
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 \( p < 0.01 \).

### 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 and Spotify (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 a user-specific weighted sum. In Figure 5, we can see that there is a 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.

In Figure 7, 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 \(\lambda = 0\), 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-DAGE, 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 attention 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 curator 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

[21] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. 2015. Improving user social media item recommendation. In W4W.