Privacy Preserving Inference of Personalized Content for Out of Matrix Users

MICHAEL SUN, Stanford University, USA
TAI VU, Stanford University, USA
ANDREW WANG, Stanford University, USA

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Information extraction**.

Additional Key Words and Phrases: datasets, neural networks, recommender systems, nlp, bert, graphs

ACM Reference Format:
Michael Sun, Tai Vu, and Andrew Wang. 2021. Privacy Preserving Inference of Personalized Content for Out of Matrix Users. In *RecSys '21: 15th ACM Conference on Recommender Systems, Sep 27–Oct 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recommendation systems in practice often rely on difficult-to-collect large-scale datasets in order to obtain a "critical mass" of user preference and content information for high-quality recommendations. With smaller datasets, the shortcomings of traditional recommendation systems begin to appear: sparsity of user preference matrices and item content lead to ineffective recommendations in data-driven systems [1, 15] particularly for cold start users and items. A large tail of niche application settings, such as the scientific and anime enthusiast communities, need recommender models catered to the needs of smaller, dynamic communities while keeping the options of privacy or anonymity.

Existing approaches fail to provide cold start recommendations while maintaining the important needs of such communities. Content-based cold start approaches [2, 14] rely on user content information, which is invasive on user personal data, and does not work effectively in the setting of new users, for whom no preferences are given. Likewise, collaborative filtering [5, 10, 16], which provides recommendations based on preference-based similarity to other users or items, struggles with cold start users and items. While there are attempts to build more advanced recommendation engines using hybrid methods [8, 19] and deep neural networks [17, 20, 22], research in alleviating cold start issues is still limited. We introduce **DeepNaniNet**, which uses a deep neural architecture to handle cold start with the option of learning rich content representations via a graph representation of the data, with edges linking users and items as well as edges linking items to items representing related show suggestions, enabling joint consideration of these data sources. We introduce neural language embeddings derived from BERT to represent both textual reviews, which constitute the edges between users and optionally item-item recommendations, which constitute the edges between items. The model is trained with WMF following previous successes[18][19]. Building on this framework, we introduce tools for generalization to new users that avoid profile mining: 1. a representation scheme of users by their "content basket", a set of a user's favorite items submitted to the service, with which to induce user representations, and 2. an

Manuscript submitted to ACM

autoencoder architecture to generalize to new users without requiring user metadata. To our knowledge, DeepNaniNet is the first deep NLP driven anime recommendation engine. We further present AnimeULike, a new dataset of anime reviews consisting of 10000 animes and 13000 users, on which the system achieves competitive user recall on both warm and cold start animes and demonstrates a learnt ability to generalize to new anime on a thematic basis. On the benchmark of CiteULike, the system achieves equivalent performance to DropoutNet's SOTA cold start numbers and avoids any performance drop upon introduction of out-of-matrix users. In realistic real world settings where half or more users are either guests or have few suggestive interactions, DeepNaniNet proves to be the superior design. On AnimeULike Warm Start, DeepNaniNet demonstrates near 7x user recall@100 over WMF and 1.5x over DropoutNet, and further experiments indicate it is learning a rich user representation space in the process.

DeepNaniNet enables high-quality recommendations in the key stress areas for niche-community recommender systems. First, it can make effective cold start recommendations for both new users and items. Second, it does so via a privacy-preserving user experience and algorithm, which is crucial for both maintaining user anonymity and effectively serving guest users. Third, it is able to jointly learn to encode diverse data sources during training for qualitatively improved recommendation quality.

2 RELATED WORK

2.1 Traditional Methods

The traditional class of solutions to building recommendation engines is collaborative filtering. Methods may model user representations via Bayesian inference of user preferences [16], unsupervised clustering of users with similar preferences [5] or latent semantic analysis [10]. Dimensionality reduction via factorization of the preferences matrix has seen success in recommendation. Weighted Matrix Factorization (WMF) has been applied towards collaborative filtering [12], and neural matrix factorizations have also been applied to recommendation [23]. Our method uses such factorization approaches to provide relevance scores to learn during training. Nonetheless, collaborative filtering approaches suffer from sparsity of user preference information. For example, user-user and item-item collaborative filtering has no remedy for cold start users and items respectively when no preferences are available. Hybrid approaches to the problem of cold start in recommendation systems integrate in both user preferences and item content. CTR [19] integrates latent embeddings for users and items with probabilistic topic modeling. CTPF [8] identifies latent topics and to form cross-topic recommendations. However, such methods introduce highly complex objective functions to incorporate additional content and preference terms. In addition, these models only handle cold start items without paying attention to cold start users.

2.1.1 Deep Learning Methods. Meanwhile, many works have applied deep learning to developing both content and collaborative filtering-based recommendation engines, such as DeepMusic [17] and RNNs [22], and CDL [20], which introduce various neural architectures to jointly learn content and preferences. Nevertheless, there have been limited attempts to address the cold start problem directly in the training procedure. DropoutNet [18] demonstrated the SOTA result in generalizing to cold start users and items by employing dropout at training time to reconstruct user-item relevance scores. However, its shortcoming is in their sharp performance decline in situations where user content is not available. The present work combines the cold start advantages of DropoutNet while avoiding existing problems of privacy invasion posed by existing user content-based approaches [3][26].

2.2 Inductive Graph Learning

Recently, general-purpose graph representation learning methods have been proposed for recommendation tasks. Pixie [7] uses biased random walks in the user-item interaction graph to generate recommendation scores, but does no learning, hence it cannot incorporate complex multimodal data end-to-end, and requires unwieldy hyperparameter tuning to design appropriate random walks for novel recommendation settings. PinSAGE [24] combines Graph Convolutional Networks [13] with efficient inference and curriculum training for web-scale recommendation. However, PinSAGE does not explicitly address the cold start problem, which we aim to do in the present work. STAR-GCN [25] was proposed to handle the cold-start setting for both users and items. In the spirit of such inductive graph learning techniques, we leverage a graph neural network (GNN) architecture component to attain item feature representations that easily generalize to the cold start setting, while able to be trained end-to-end with other multimodal data sources. In particular, we make use of a GNN architecture based on GINEConv [11] to obtain node embeddings that encode textual information features, corresponding to reviews, along graph edges. In contrast to general-purpose graph representation learning methods, however, DeepNaniNet uses matrix factorization techniques which are more tailored to recommendation tasks, leading to superior performance as demonstrated in the experiments.

3 DATASET

3.1 AnimeULike

We propose a new dataset of anime reviews and ratings for recommendation. The dataset represents an impactful real-world recommendation setting which benefits a substantial niche community; it also carries key challenges of serving cold start animes and out-of-matrix users, thus serving as a benchmark for the generalizability of recommendation systems given sparse training data, with regard to privacy constraints. Here we describe the key properties and construction of the dataset.

Collection Pipeline. We crawl a popular website of anime reviews and ratings, MyAnimeList.net, to obtain a rich graph of users and animes ("items") as well as textual reviews and synopses. Out of respect for MyAnimeList.net's community, we avoid scraping user profiles directly. Instead, we crawl the listings of top rated animes to discover users, and additionally only save their id's to disk.

User-Item Reviews. We obtain a graph of users and animes (items), connected by textual and numerical reviews. Our pipeline crawls through each of the site's top 10000 rated animes (as of Feb 2021). For each anime, we retrieve the numerical features and synopsis from its "profile page" and crawl its "reviews page", retrieving all available reviews, each of which comes with a written body, reviewer username, and rating (truncating at page 50 on the reviews to avoid spam). The list of users in the dataset is thus the set of all reviewers discovered this way. We send all users encountered during this phase to an additional sub-pipeline that crawls their past reviews for additional ratings. This way of extracting ratings from written reviews ensures ratings' integrity and retrieval of text that is directly responsible for predicting ratings. This collection step populates a preference matrix $R \in \mathbb{R}^{N \times M}$ whose rows are users, columns are items and entries are numerical ratings from a given user to the given item.

Item-Item Recommendations. To retrieve recommendations between a pair of animes, we crawl the "userrecs" tab for each of the 10000 animes' pages, representing written recommendations between two animes submitted by users, then concatenate all recommendations between each pair of animes into one body of text (made by k recommenders — we include k as a metadata attribute `num_recommenders`). Those with `num_recommenders=1` are ignored, lest they be

Table 1. Anime dataset specification with examples. The item-item anime graph has 27266 edges for 10000 animes.

Section	Description	Specification	Example Entry
User-Anime Preferences	Rating (1-10, 0 for missing); (optional) textual review	$\mathbb{R}$	8; "First of all, I have seen the original FMA and although it was very popular and original, the pacing and conclusion did not sit too well with me…"
Anime-Anime Suggestions	Concatenation of all written recommendations made from anime1 to anime2	String	"**Hunter x Hunter (2011)** Both are the best shounen in the world to me! They start off with adventures of a brave main character and go on into darker themes. … **Shingeki no Kyojin** …"
Anime features	Numerical Features (avg. rating, avg. rating (rounded), #ratings, rank, #members, #favorites)	$\mathbb{R}^6$	(9.18, 9, 1, 3, 2277948, 185915)
Anime Textual Content	Synopsis; Concatenation of all written reviews	String	"'In order for something to be obtained, something of equal value must be lost.' Alchemy is bound by this Law of Equivalent Exchange…"
User ratings	Anime ratings by given user	$\mathbb{R}^{\lvert U^{\text{train/val/test}}\rvert}$	$[9, 0, 0, \ldots, 8, \ldots]$
User content basket	Set of animes a user likes	Set<Item Ids>	{Fullmetal Alchemist: Brotherhood, Hunter x Hunter, Code Geass} (id mapped)

unsubstantiated or spam. This collection step results in a graph whose nodes are items and edges are annotated with textual recommendations between pairs of items.

Features. Each anime page also comes with rich metadata and textual features. For each anime, we retrieved its MAL ID, numerical features (average rating, popularity, rank, members, favorites), and all reviews concatenated as one string. In total, we collected 27266 anime pairs (each a long text body). Each (anime 1, anime 2) pair comes with all recommendations written on anime 1's page for anime 2 (an undirected edge). Our entire pipeline is parallelized across both sub-pipelines and runs in 20 mins. We also make the discovery and collection pipeline to be configurable. Our final user-item graph for training only contains an id for each anonymized user id's and the numerical rating (0-10) given. Our compiled dataset and web crawling codebase will be released fully upon deanonymization.

Preprocessing. We first randomly split our dataset of 10000 top-rated anime shows into training, validation and test with a $8:1:1$ ratio, resulting in M^{train}, M^{val}, and M^{test} animes in each split respectively. We apply weighted matrix factorization (WMF) to approximate the training user-item rating matrix $R^{\text{train}} \in \mathbb{R}^{N \times M^{\text{train}}}$, as a matrix product $R_{u,v} \approx U_u^{\text{train}} (V_v^{\text{train}})^T$, where the rows of U^{train} and V^{train} are dense latent representations of the N users and M^{train} train set items, respectively.

3.2 CiteULike

In addition to our anime dataset, we tested our system on the CiteULike database [4]. This dataset contains 5551 users and 16980 articles, and each user has, on average, 37 articles. We demonstrate that DeepNaniNet achieves superior performance on widely-used benchmark CiteULike in the setting of out-of-matrix users, while preserving performance

in the in-matrix case compared to the state of the art. To obtain item content ϕ_v^V, we follow DropoutNet [18] and run SVD on the TF-IDF top-8000 matrix, with 300 components, on the associated documents for CiteULike.

4 DEEPNANINET: RECOMMENDATION FRAMEWORK

We propose a neural recommender system architecture, whose goal is to learn rich user and item representations that can predict user-item preferences given possibly incomplete, heterogeneous information (in the form of item and user content, or a partial view of user preferences). Within this system, we propose a "user content basket" technique for making accurate recommendations to users requiring minimal personal information, as well as an embedding dropout technique for improved out-of-matrix generalization. Together, these components form a robust system for generalizable recommendations on smaller, heterogeneous datasets.

Preliminaries

We define the latent vectors obtained by WMF as $U_u \in \mathbb{R}^h$ for the latent vector of user u and $V_v \in \mathbb{R}^h$ for the latent vector of item v. Let $V(u)$ be the set of all items user u has interacted with $U(v)$ be the set of users an item v has received ratings from. We presume that each item has an associated content vector denoted ϕ_v^V. Depending on the dataset, each user may come with an associated content vector denoted ϕ_u^U too.

Architecture

The architecture consists of a user encoder and an item encoder. Each encoder takes in a list of known user-item preferences U_u and V_v respectively, as well as a representation of any associated content (textual reviews or item-item suggestions), and outputs a latent embedding for the given user or item.

User and Item Encoder. For each user u, the user encoder takes in its latent embeddings U_u from the user-item preferences matrix, as well as its user content vector Φ_u^U (if provided), and outputs an encoder embedding $\hat{U}_u \in \mathbb{R}^r$, as shown in green in Fig 1. The encoder proceeds as follows dropout. The user preferences vector U_u is passed through a Dropout layer (rate P_u). The resulting user preferences vector after dropout is given by $\tilde{U}_u$ (see below). The user preferences vector after dropout $\tilde{U}_u$ is fed into a neural network layer $f_U\left(\tilde{U}_u\right) := \tanh\left(W_U\tilde{U}_u\right)$ to produce an intermediate representation, and user content Φ_u^U is fed into a neural module $f_{\Phi^U}\left(\Phi_u^U\right) := \tanh\left(W_U^\Phi\tilde{U}_u\right)$. The feature representations are concatenated and fed into a final layer $f_{\mathcal{U}}$: the output is given by:

$$\hat{U}_u = f_{\mathcal{U}}\left(\left[f_U\left(U_u\right); f_{\phi^U}\left(\phi_u^U\right)\right]\right) := \tanh\left(W_{\mathcal{U}}\left[f_U\left(U_u\right); f_{\phi^U}\left(\phi_u^U\right)\right]\right) \tag{1}$$

where $[;]$ denotes concatenation. Similarly, for each item v, we pass V_v and Φ_v^V to obtain $\hat{V}_v \in \mathbb{R}^r$, as shown in blue in Fig 1. Following prior work [18], we standardize U, V and apply batch normalization after $f_U, f_V, f_{\phi^U}, f_{\phi^V}$ with 500 output units for f_U, f_V and rank $r = 200$ output units for f_{ϕ^U}, f_{ϕ^V}.

Content Encoder. We experiment with choices for the content encoder ϕ^V. When the content is textual (CiteULike documents or textual anime reviews), we take these functions to be either a BERT encoder or SVD on the tf-idf scores. Both methods take in a textual document and output a vector corresponding to that document, which we take as ϕ^V. For BERT, we carry out a comparison of pretraining and finetuning strategies in the experiments to identify the most effective method for generalization to niche textual datasets.

In particular, for the BERT encoders, we leveraged the pretrained weights of BERT in HuggingFace [21] and finetuning it on our downstream task. In addition, we domain adapted our BERT models on all the texts in our dataset using masked language modeling [6], and then used these in-domain transferred models as our content encoders, with additional finetuning on the AnimeUReallyLike subdataset.

When the content is a graph $G = (V, E)$ (item-item suggestions for anime), with associated node feature matrix X, we compare the following graph neural networks (GNNs) to provide the content vectors Φ_v:

- Graph Convolutional Network [13]. The GNN takes in (G, X) and a given item v and outputs an embedding for the item, $f_{\phi^V}\left(\phi_v^V\right) := \hat{A}\text{ReLU}\left(\hat{A}XW_0\right)W_1$ for a two-layer GNN, where W_0 and W_1 are learnable parameters and $\hat{A}$ is the adjacency matrix after preprocessing according to [13].
- GINE [11]. When there are additional features $V_{u,v}$ for each edge $(u, v) \in E$, we use GINE, which takes in (G, X, V) and a given item v and outputs an embedding for the item, which we take as $f_{\phi^V}\left(\phi_v^V\right)$. GINE performs multiple updates of the form:

$$x_i^{k+1} \leftarrow \text{ReLU}\left(h\left(x_i^k + \sum_{j \in N(i)} \text{ReLU}\left(x_j + V_{j,i}\right)\right)\right) \tag{2}$$

  where x_i^k is the embedding for the item after each layer k, h is a neural network layer and $N(i)$ are the neighbors of node i in G. We take $f_{\phi^V}\left(\phi_v^V\right) = x_v^K$ after a fixed number of GNN layers K.

Since availability of (item, item) edge content is contingent on the dataset, we concactenate $f_{\phi^V}\left(\phi_v^V\right)$ to the inputs to f_{ϕ^V} and later explore its ablation.

Relevance Score Prediction. Given user u and item v, the goal of the model is to predict the relevance score between the two. Given the output of the user and item encoders, we efficiently compute this score as $\hat{U}_u^T \hat{V}_v$, as shown in fig 1.

Our Solution for Cold Start

For guest users (for whom U_u is not available), we set $U_u = 0$ and train our model to periodically "drop out" U_u by relying on ϕ_u^U instead. Once a few interactions are collected, the user transform approximation $U_u \approx \frac{1}{|V(u)|}\sum_{v \in V(u)} V_v$ is taken instead - and vice-versa for cold start items. This is the approximation used by [18] before each re-computation of WMF. We propose a new solution via an extension of the model's architecture instead.

User Content Basket. One key insight is to model the user content as the average of its user's corresponding items:

$$\phi_u^U = \mathbb{E}_{v \in V(u)}\phi_v^V \approx \frac{1}{|V(u)|}\phi_v^V. \tag{3}$$

This representation is advantageous from a modelling perspective because ϕ^U, ϕ^V are in the same latent space, making the model more compatible with the autoencoder-inspired objective. This design decision also eliminates complications in user experience, with no privacy invasion from social profile mining, implications for user churn rate via soliciting for preferences or reduced service quality from exploration approaches that are difficult and expensive to implement. In particular, "guest users" who do not have prior information in the system can voluntarily submit a "content basket" $V(u)$ consisting of a small set of items that they prefer for inference.

Learning to Generalize. At training time, we explicitly train the model's ability to generalize to new items by performing one of three possible options:

- Leave all embeddings as is.
- User Dropout: $\left(V_v, \phi_v^V\right) \rightarrow \left(\text{mask}(V_v), \phi_v^V\right)$ (with probability `user_drop_p`).
- Item Dropout: $\left(U_u, \phi_u^U\right) \rightarrow \left(\text{mask}(U_u), \phi_u^U\right)$ (with probability `item_drop_p`).

We note [18] instead trains for cold start via training with both dropping out U_u and the "user transform" $(U_u, \phi_u^U) \rightarrow \left(\frac{1}{|V(u)|} \cdot \sum_{v \in V(u)} V_v, \phi_u^U\right)$ and vice-versa "item transform" for out-of-matrix (unfactored) items. our solution has no need for this as inference is made solely with ϕ_u^U instead, with no loss of performance in reconstructing $U_u^T V_v$. We perform element-wise masking on (`user_drop_p` and `item_drop_p`) samples per minibatch to train for cold start (despite $V_v^{\text{val/test}} = 0$ and/or $U_u^{\text{val/test}}$ in that setting). We observe improved cold start generalization this way with mask = Dropout(drop_p), which contrasts with [18]'s mask = 0. Later, we apply mask$(U_u) = U_u$+Gaussian Noise $(\mu(U), \sigma(V))$ to how these masking choices reveal DeepNaniNet's characteristics of a denoising autoencoder.

Training

To train the model, we minimize the loss:

$$L = \sum_{(u,v) \in S} \left(U_u^T V_v - \hat{U}_u^T \hat{V}_v\right)^2 \tag{4}$$

via stochastic gradient descent, where S is a set of training users and items. Intuitively, the learned embeddings should be able to reconstruct those learned by WMF, hence achieving the recommendation quality of WMF when full preferences are available. However, unlike WMF, the generalization ability of the neural framework will allow the model to make predictions about users not seen during training, as will be demonstrated in the experiments.

We note that the loss function is expensive to compute in practice, due to summing over $O(NM)$ terms. Hence, we approximate the loss via negative sampling, where S consists of k positive examples (pairs (u, v) with nonzero preference matrix entry) and $5k$ negative examples (pairs with zero entry).

Throughout all experiments, we keep the same hyperparameters as in DropoutNet [18] (e.g. dropout of 0.5), including the model architecture (single hidden units of 500 units), layer architecture (linear + batch normalization + tanh), and training data (as per trained WMF features and train/validation splits for CiteULike cold start), for consistency of evaluation. We follow the same practice of batching by users for each user sampling a fixed number of items, but differ in that we make sure each pass sees all true positive interactions. Unlike DropoutNet, we train without momentum for batch SGD, which we found to work better given this batching scheme. We sample random items at a ratio of 5:1 positives to negatives per epoch, ensuring each user sees a diverse set of item candidates. This setup achieves superior performance on AnimeULike while reproducing reported results on DropoutNet.

5 EXPERIMENTS

We run several experiments on both the anime and scientific article domains to understand generalization performance in terms of cold start and out-of-matrix users. Furthermore, we demonstrate robustness of the model to corruption rate. We run ablations to demonstrate the efficacy of our encoder components (GNN and domain-adapted BERT representations) in representing ϕ^V and ϕ^U. Finally, we qualitatively analyze the method's recommendations in the case of AnimeULike and AnimeUReallyLike.

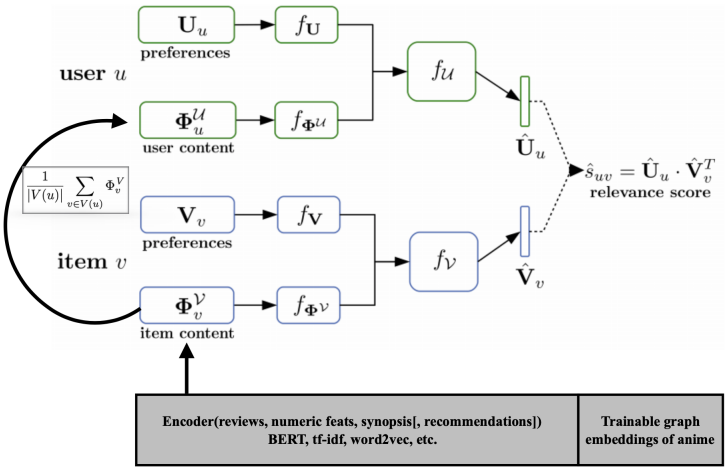


Fig. 1. DeepNaniNet: Item content featurized by concatenating encoding with graph embedding (Edited on top of DropoutNet paper's diagram [18])

5.1 Evaluation method

We adopt user-oriented recall@K as our default quantitative metric (reported results set K=100 to stay consistent with prior work[18][19]) and analyze qualitative patterns in a later section. We evaluate methods along two main scenarios.

- In the *warm start* setting, the system has observed the preferences U_u for the user to retrieve recommendations for (resp. V_v for items). Optionally, the system can incorporate user content ϕ_u^U (resp. ϕ_v^V for item content).
- In the *cold start* setting, the system has observed no preferences for the user to retrieve recommendations for (i.e. $U_u = 0$, resp. $V_v = 0$ for items) and thus must rely solely on content ϕ_v^U (resp. ϕ_v^V). In cases where ϕ_u^U (or ϕ_v^V) is not available, inference would not be possible. To address this, past approaches have proposed approximations of U_u as the user's preferences become available. Instead, DeepNaniNet represents users via their content baskets (see Techniques for Improved Generalization), enabling effective inference in this setting from the very start.

5.2 Preprocessing

For the anime dataset, we split the 10000 animes into train-val-test via a 8:1:1 split. We apply WMF on the training preference matrix (of size 12767×8000), excluding users who did not rate any items in the training set, to obtain a latent user matrix U^{train} and latent item matrix V^{train}. For warm start, we include 1000 validation animes (leaving out 1000 for future experiments) and follow the same procedure as[18][19] to refactorize a 12767×9000 matrix and test on fold 1 (all cases nearly the same). We explicitly prevent corner cases where a user may be left with no animes to form his/her content basket. On average, the size of the user's content basket is 5.3 in the cold start train fold, 0.6 in the cold start val fold, 4.8 on the warm start train fold, and 1.1 on the warm start validation fold.

We now define some important hyperparameters. For DropoutNet, we define `user_transform_p` as the rate by which we substitute user vectors via a "user transform" $U_u \approx \frac{1}{|V(u)|}\sum_{v\in V(u)} V_v$ (abbrev. "UT" in experiments), its proposed approach to handling out-of-matrix users. We drop out user embeddings in U with probability `user_drop_p` and item embeddings in V with probability `item_drop_p`. For DeepNaniNet, which utilizes the content basket representation, we take $U_u = 0$ for out-of-matrix users at inference time (hence such users are represented entirely by ϕ_u^U).

5.3 Experiments

5.3.1 CiteULike Cold Start. The user transform (UT) approximation used in DropoutNet, while resulting in performance gains for out-of-matrix users, causes a a significant performance drop for in-matrix users, hence cannot serve both classes of users simultaneously. This suggests that DropoutNet overfits to U, and approximations of the user representation U_u (even by a good approximation since $|V(u)| \approx 30$ for CiteULike), poses a significant handicap. Poor out-of-matrix performance also has negative consequences for industrial models, necessitating that they be retrained entirely after each computation of WMF as U and V shift due to evolving distribution of users and documents over time.

Table 2. On recommendation for CiteULike, DeepNaniNet gracefully handles both the in- and out-of-matrix cases, achieving comparable or favorable performance in both cases. In contrast, DropoutNet is not able to handle both cases simultaneously, even with the user transform technique for handling out-of-matrix users, due to the overfitting to U.

	In-matrix	50%-50%	Out-of-matrix
DN (no UT)	**62.1**	59.1	56.7
DN (UT)	7.7	30.9	55.4
DNN	61.3	**61.2**	**61.0**

In realistic real world settings where up to half of users could be guests, this limitation implies the impracticality of serving guest or newly registered users (for whom there are no or few interactions) without either a) recomputing WMF after sufficient exploration or b) finding more invasive ways to obtain better user representations. While DropoutNet holds onto a slight edge in the case of all in-matrix users (despite DeepNaniNet having access to U all the same), we see this as a positive indication that DeepNaniNet is instead learning a rich latent space beyond merely information contained in U or V.

5.3.2 AnimeULike Warm Start. We next compare anime recommendation performance in the warm start setting. For AnimeULike, we keep the same setup as before. We experimented with both TF-IDF and transferred BERT as the content encoder. Our base model experiments reveal transferred BERT as the better ϕ^V anime2vec encoder for *both* DropoutNet and DeepNaniNet (whereas in the CiteULike domain, TF-IDF has the edge). Thus, to put the best foot forward for both models, we fix the domain-adapted BERT as the anime2vec encoder for all warm start AnimeULike experiments.

Table 3. AnimeULike: Warm Start. DeepNaniNet surpasses DropoutNet for both in-matrix and out-of-matrix users. In brackets are preliminary runs with tf-idf as the content encoder.

Method	In-matrix	Out-of-matrix
WMF	9.4	8.1
Popularity sort	N/A	2.4
Random guessing	N/A	1.1
DN	38.3[/37.6]	36.7[/36.4]
DN (UT)	37.5	36.9
DN (added GNN)	42.0	39.2
DN (added GNN, UT)	41.0	39.1
DNN (removed GNN)	**64.3**[/61.0]	**61.4**[/60.9]
DNN (V=0)	63.3	61.1
DNN (full)	64.2	59.7

Our initial observation is the magnitude-fold improvement over the WMF baseline. In [18], only a 0.001 improvement is made over the WMF baseline on warm start. The poor metrics of WMF warm start recommendations seems consistent with users' experiences using our demo, with one user source commenting, "I typed in Flip Flappers and it just gave me the FMAB, AOT S3.5, and AOT S4 despite them being the exact opposite of what I'm looking for. Looks like it just picked the top 3 rated shows from MAL." As the *popularity sort* baseline shows, correlation with popularity is low, so methods like WMF and CF that learn off of preference overlap between users perform poorly. The sparsity (5 items/user for warm start) and stochasticity of user anime interactions produces

much poorer approximations of U and V than as in CiteUlike, forcing the autoencoder to rely on content instead. This is substantiated by the fact DeepNaniNet jumps to *over double* the performance of DropoutNet from the very start. We run each model the same number of updates until performance plateaus. While DropoutNet makes up for a little differential, it's clear by now using content baskets is the superior design. Visually, there's a large separation between the two families of models' performances. Adding GNN to DN helps some, but the performance bottleneck is most likely the expressiveness of U, hence leading us to conclude our performance owes itself to the content basket representations. This suggests our model is constructing a rich shared representation space between (ϕ^V, ϕ^U) and (V, U) from the start, exhibiting the ideal behavior of a denoising autoencoder and thus avoiding the discussed pitfalls of overfitting to U or V. In fact, we observe swapping $V_v = 0$ at evaluation time not only doesn't hurt performance but increases it marginally.

Table 4. DeepNaniNet is robust to noise corruption of up to ≈ 70% samples.

Corruption rate	
0.1	62.7
0.3	63.1
0.5	64.0
0.7	64.3
0.9	63.3

To further explore this hypothesis in isolation, we ran experiments *corrupting* inputs U (by Gaussian noise of equal mean and std) and confirm a pattern: not only does corrupting U not hurt performance, but seems to *boost* performance, with a 0.7 corruption rate equivalent to our best model overall. This suggests the autoencoder quickly discerns U_u is mostly noise, and adapts faster to the rich representational space constructed off of our deep encoders, resulting in a higher performance in the end. At the same time, too much corruption reveals diminishing returns: the autoencoder still requires partial observability of U to bridge the latent spaces.

Our only surprise is that incorporating graph representations didn't seem to help, hence the relational information in the anime-anime suggestion graph may be more useful for recommending new animes rather than established ones, as will be demonstrated in the cold start setting.

5.3.3 AnimeULike Cold Start. Next, we demonstrate the improved generalization ability of DeepNaniNet in the cold start setting, particularly in handling out-of-matrix users.

We observe superior performance of DNN (REMOVED GNN) which omits the GNN representation of the item-item graph, surpassing DropoutNet. As established in the previous section, we confirm content baskets to be the superior design for cold start as well. When trained with user transform to handle out-of-matrix users, DropoutNet deteriorates on in-matrix users as a result and fails to deliver the intended boost on out-of-matrix users. DeepNaniNet, on the other hand performs better for **both** in-matrix *and* out-of-matrix user representations, whereas on CiteULike it just maintains in-matrix performance. Most intriguingly, it appears on cold start items, both DropoutNet and DeepNaniNet prefer their respective out-of-matrix representations of users (user transform[1] and $U_u = 0$, respectively). We discuss the fascinating implication

Table 5. AnimeULike Cold Start. DeepNaniNet achieves superior out-of-matrix recommendation performance on anime recommendation while maintaining high in-matrix performance.

Method	In-matrix	Out-of-matrix
Popularity sort		13.8
Random guessing		10.0
DN (no UT)	44.1	48.5
DN (UT)	42.6	48.4
DNN (removed GNN)	47.9	52.0
DNN	**55.1**	**56.5**

[1]It may be the case on AnimeULike, U_u is basically noise already, so DropoutNet can't overfit to U, causing the user transform approximation to help to an extent.

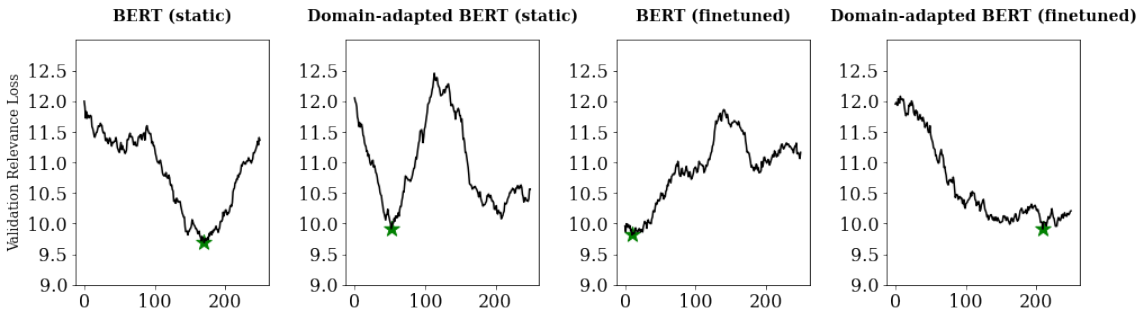
of this in a later section. Our further experiments with GNN on out-of-matrix user representations demonstrated the best results of all: 0.565 user recall@100 which highlight the true advantage of jointly learning anime representations via our item-item graph. We believe this is due to GNN's ability to inductively generalize to new users. In particular, if new users have connections to similar items as users in the training set, the model is inductively biased towards making similar predictions for such users [9]. In summary, DeepNaniNet's superior performance across both in-matrix and out-of-matrix users suggest it has constructed a richer user representational space, separate from U, from which it can better reconstruct relevances in the original space, in conjunction with rich anime content ϕ_v^V.

5.3.4 AnimeUReallyLike. Lastly, we experiment with finetuning BERT as part of an end-to-end model. This implies end-to-end backpropagation to parameters of ϕ^V (and thereby ϕ^U) via sampled content baskets.

We take a small, selected sample of 10287 (user, item) training entries and 5585 validation entries from AnimeULike that are thresholded with true relevances of >10 or <-6. We denote this small dataset of "extreme" relevances (either a user really hates or loves an anime) as **AnimeUReallyLike** and train differential language encoders for the challenging task of predicting these "extreme" preferences. Due to the resource constraints of training BERT, we leave out recall as it is not competitive and instead on preliminary findings with the validation mean square error relevance loss.

Table 6. Loss curves are running averages over 50 batches to account for sampling stochasticity, trained with Adam optimizer. We found this sampling strategy achieves lower loss. Table losses are lowest achieved averages (green star in plot). For the finetuned models, we impose a 0.99 learning rate decay on BERT's unfrozen layers initialized at 1e-4.

Models (fix drop_p=0.75)	MSE 4 (drop_p = 0)	MSE (drop_p = 0.25)	MSE (drop_p = 0.5)	MSE (drop_p = 0.75)	MSE (drop_p = 1)
bert/bert_static, drop_p = 0	10.363	9.947	10.628	9.689	10.716
bert/pretrained_bert_static, drop_p = 0	10.034	10.480	10.310	9.915	10.207
bert/bert_finetune, drop_p = 0	10.304	10.446	10.134	9.816	11.029
bert/pretrained_bert_finnetune, drop_p = 0	10.595	9.934	10.519	9.913	10.183



All encoders report lower losses when drop_p=0.75 as opposed to 0.5, attesting to the richer representations being learned via **end-to-end** backpropagation of sampled content baskets to content representations. Relatively, the loss curves suggest finetuning our domain-adapted BERT to be the superior choice for downstream training.

While static BERT reports the single lowest value, downstream layers immediately overfit due to the language encoder's out-of-distribution representations. This can thus complicate attempts of online learning that periodically

Table 7. Transfer-finetuned BERT anime2vec: (node, edge) encoder choices, (AnimeUReallyLike)

(drop_p=0.5/drop_p=0.75)	TF-IDF	BERT static	BERT finetuned	BERT transfer static	BERT transfer finetuned
TF-IDF	(9.721/9.953)	(9.911/9.863)	(10.247/8.883)	(9.424/9.586)	(10.070/10.265)

Table 8. To test the ability to reconstruct *extreme* relevances (i.e. a user's favorite show, or an item's most fervent rater), we consider the metrics of items/user MRR and users/item MRR relative to baseline encoders.

Model	TF-IDF	TF-IDF + GNN	TF-IDF + EdgeConv	Static BERT	Fine-tuned BERT	Static Pre-trained BERT	Fine-tuned Pre-trained BERT
MRR (Items/User)	2.2e-4 (2.5e-4)	2.3e-4	3.2e-4	2.2e-4	2.5e-4	**4.0e-4**	**5.2e-4**
MRR (Users/Item)	4.1e-4 (1.6e-4)	3.6e-4	8.7e-4	**11.5e-4**	**8.7e-4**	2.6e-4	2.3e-4

update our model to adapt to distributional shift. Meanwhile, fine-tuning BERT directly causes too large a domain shift in ϕ^V for tuning downstream layers (tested by independent experiments with separate learning rate schedulers for BERT layers). Instead, we advocate for finetuning our domain-adapted BERT, which consistently reports low variance and the smoothest validation loss curve across runs, as shown. Settling on the domain-adapted BERT as our anime2vec encoder, we ran experiments over all node feature encoders as well, and observed consistently lower losses with TF-IDF as the node encoder. We suspect this to be a form of feature fusion where TF-IDF features supplements the fine-tuning BERT anime2vec.

The expected result of *random guessing* is in parentheses. Despite seeing only a small subset (4592 positive, 6235 negative entries), these models fare multiples better than random chance. Transferring BERT in-domain is advantageous for retrieving a user's top show due to pretraining on a diverse set of high quality user reviews. Moreover, the BERT models outshine TF-IDF in at least one MRR department. We suspect that while TF-IDF is a better comprehensive encoder in non-semantic domains (i.e. CiteULike) due to high correlation in features, it struggles to find a decision boundary in semantic domains where most tokens appear in both extremely positive and extremely negative cases. Thus, a second case for deep language encoders is their complex non-linear decision boundaries that excel in semantic settings. We discuss these advantages further in the section below.

6 DISCUSSION

6.1 Qualitative Analysis

In our app, the user experience using **DeepNaniNet** with WMF and top-K CF feels drastically different. Users have complained WMF only returns what's already popular. While top-K CF's recommendations feel a bit more intentional (as adopted by the best attempts to build anime recommendation engines like here), it achieves little more than recommendations you may get after consulting multiple friends. Our approach consistently captures underrated anime (ones that fly under the radar but when you look into its synopsis, is thematic), making its high recall numbers all the more impressive. As an example, our prior prototype (which recommended exclusively from the top 250 shows for more traction) produced the following when querying with two highly regarded shounens.

["Fullmetal Alchemist Brotherhood" (FMAB)[2], "Shingeky No Kyojin" (AoT/Attack on Titan)] ⇒

- WMF: 1) Haikyuu, 2) Hunter x Hunter [3]
- Top-K: 1) Mushishi, 2) Stein's Gate, 3) Haikyuu, 4) Hunter x Hunter [4]
- DeepNaniNet: 1) Hajime no Ippo [5], 2) Nana[6], 3) Gintama [7]

We make the following generalizations: WMF is biased towards popular anime. Top-K is biased towards highly rated anime with lots of item-item recommendations, while **DeepNaniNet** seems capable of *actually reading* information on other shows and outputting thematically similar (and often underrated) shows. Perhaps the biggest advantage of DeepNaniNet comes from the fact it actually represents all available information word-for-word to make recommendations. This avoids content popularity biases and consistently outputs underrated shows. However, it also comes with high sensitivity to strong sentiment and word-level mentions, esp. on AnimeUReallyLike. An example is when running the AnimeUReallyLike model on the same query set ['Fullmetal Alchemist Brotherhood', 'Shingeky No Kyojin'] (FMAB and AoT), two of the highest rated/regarded shounen series of all time. We observe:

- TF-IDF GNN: A Farewell to Arms[8], Armored Trooper Votoms[9], Hotori[10], Sailor Moon SuperS the Movie
- TF-IDF GINE: Armored Trooper Votoms[11], A Farewell to Arms, Slayers, Patlabor: The Movie
- Finetuned BERT: Royal Space Force[12], Desert Punk, Ginga Eiyuu Densetsu[13], Mobile Sui Gundam[14], Azur Lane[15]
- Fine-tuned transferred BERT: .hack//Sign, .hack//Quantum, Fate/Zero, .hack//Liminality, .hack//Gift

We observe GNN GINE both converge on a similar set of results that all common themes of military, humanity's war, revolution, etc. Reading FMAB and AoT's descriptions, we see these themes present in their synopses: "military allies, colonel, lieutenant, nationwide conspiracy, state, law" and "humanity, extinction, defensive barriers, fight for survival, Survey Corps, military unit, brutal war, walls". Fascinatingly, we observe plot-level similarities with FMAB and Hotori. FMAB is about two brothers who lost parts of their physical bodies. "It is the hope that they would both eventually return to their original bodies..." whereas Hotori has this identical element.

TF-IDF seems capable of capturing similarities from narrow dimensions (such as shallow mentions of military themes), even if those dimensions are not the central themes of either FMAB or AoT.

[2]FMAB is rated #1 on MAL all-time and AoT is trending #1 in popularity.
[3]Both Haikyuu and Hunter x Hunter are among the most popular shounen.
[4]Mushishi is irrelevant in content but has *a lot* of item-item edges (preferred by CF) hence a diverse rater base. Stein's Gate is highly regarded but not action/adventure shounen, again demonstrating the popularity bias.
[5]Highly underrated shounen!
[6]A lesser known slice of life, has a synopsis that says two girls travelling together in search of one girl's boyfriend (note: FMAB's synopsis talks about two brothers travelling together in search of the philosopher's stone!)
[7]A bit of an offbeat satirical shounen that pokes fun at other shounens!
[8]A Farewell to Arms: a story of "power suit-wearin' men tasked with disarming automatic tanks in a post-apocalyptic Tokyo"
[9]Armored Trooper Votoms: set in "a century of bloodshed between warring star systems... flames of war..." where "a special forces powered-armor pilot is suddenly transferred into a unit engaged in a secert and highly illegal mission to steal military secrets..." (you get the idea)
[10]Hotori: At the Personality Plant, robots are being built and slowly outfitted with the artificial memories of real people." The main character, Suzu, "is one such robot."
[11]Armored Trooper Votoms: set in a city "built form the labors of mechanical beasts... with incredible destructive power as a new type of advanced weaponry"
[12]Royal Space Force: Protagonist is part of the country's space force, who embark on a mission to redeem humanity by restoring its strength
[13]Ginga Eiyuu Densetsu: About a coup staged by the National Salvation Military Council under the direction of the Galactic Empire, happening during civil wars in both the Alliance and the Empire
[14]Mobile Sui Gundam: About a space immigrant who joins the League Militaire, a militia frustrated with their empire's cruelty, who fights to bring an end to the Zanscare Empire's reign
[15]Azur Lane: pits "a divided humanity" which "stood in complete solidarity" against "an alien force with an arsenal far surpassing the limits of current technology"; with countries joining forces, "paving the way for the improvement of modern warfare"... during "neverending conflict within humankind" (basically if FMAB and AoT had a baby... this would be it!)

Fine-tuning BERT, meanwhile, **captures similarity across more complex dimensions**. The shows are more diverse in plot while rooted in common themes across militant conflict, failure of government, humanity, and revolution.

At the same time, this turned out as an adversial example for the transferred BERT encoder. We discovered the synopses and reviews for shows within the ".hack" franchise (aptly named) repetitively refer to the series as a whole, making the language encoder sensitive to its mentions as opposed to learning each show's distinct content, resulting in the four of them to be recommended together. Looking past that, Fate/Zero is a *very* good recommendation[16]. Nonetheless, this adversarial example poses a concern of using deep language encoders four our system. We are motivated to pursue, as a direction of future study, the effect on recommendations due to the encoder's sensitivity to word mentions or extreme forms of sentiment.

6.2 Applications

Our system is extremely practical for inference: only $\phi^V, \hat{U}, \hat{V}$ needs to be cached (only the latter two for in-matrix users) and an additional step of computing $\hat{U}$ off of ϕ^V for guest users with the option of approximating $\hat{U_{guest}} \approx \frac{1}{|V(u_{guest})|} \sum_{v \in V(u_{guest})} \hat{V}_v$. This enables parallelism for faster retrieval. This is a huge win for SaaS recommendation services bootstrapping off minimal user data. We envision the flourishing of open-source representations for popular items across popular culture and media, enabling more niche services to experiment with ways of content basket design for satisfying more domain-specific tastes. Guest users can then experiment and enjoy high-quality recommendations with these services without fear of being mined of personal data.

For large companies, this can avoid many of the privacy concerns and technical pains of storing, managing, and exploiting a customer's entire lifecycle on the application. As companies adopt more content-based recommendation systems, we believe *latent* modelling of ϕ^U conditional on context (i.e. $\phi^U_u = \sum_{v \in V(u)} f(v|u,c)\phi^V_v$ weighted content baskets) can design more intentional recommendations (as in Spotify's user explanatory framework) dependent on a user's context c (i.e. "looking for fantasy" or "pumped up").

Due to our strong performance on cold start, we believe content creators and advertising channels can use services built off this model to test potential audience traction with new types of content, whose deep representations avoid the negative feedback loop of collaborative-filtering approaches.

7 CONCLUSION

We introduced DeepNaniNet, a neural recommender system framework for reconstructing user-item preferences via rich content encodings, and our techniques for better cold start generalization. We replicated DropoutNet's SOTA cold start results on CiteULike, where our model maintains equally strong performance across the out vs. in-matrix users, hence outperforming DropoutNet in the realistic real-world setting where 50% of users are guest users. We introduced AnimeULike, a dataset rich in content but sparse in preferences, and demonstrated strong performance on both warm and cold start - notably a 7-fold improvement over the WMF baseline - including further experiments revealing rich properties of a denoising autoencoder. We demonstrated further gains in generalization to cold start items via jointly learnt graph representations. Finally, we made the case for deep, differentiable language encoders and feasibility for end-to-end training. Lastly, we close with our motivation that started it all: to deliver more meaningful, personalized, and engaging content for users (whether old, new or guest) without compromising our principles for user privacy.

[16] Fate/Zero: not only thematically similar (war, battle royale, etc.) but is regarded as a crossover between both FMAB and AoT: exploration of deep themes and unapologetic cruelty

REFERENCES

[1] Sajad Ahmadian, Mohsen Afsharchi, and Majid Meghdadi. 2019. A novel approach based on multi-view reliability measures to alleviate data sparsity in recommender systems. *Multimedia Tools and Applications* 78, 13 (2019), 17763–17798.

[2] Lucas Bernardi, Jaap Kamps, Julia Kiseleva, and Melanie JI Müller. 2015. The continuous cold start problem in e-commerce recommender systems. *arXiv preprint arXiv:1508.01177* (2015).

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

[4] Toine Bogers and Antal Van den Bosch. 2008. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM conference on Recommender systems*. 287–290.

[5] Sonny Han Seng Chee, Jiawei Han, and Ke Wang. 2001. Rectree: An efficient collaborative filtering method. In *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 141–151.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 world wide web conference*. 1775–1784.

[8] Prem Gopalan, Jake M Hofman, and David M Blei. 2013. Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704* (2013).

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[10] Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 89–115.

[11] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HJlWWJSFDH

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[14] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold start problem in recommender systems. *Expert Systems with Applications* 41, 4 (2014), 2065–2073.

[15] Monika Singh. 2020. Scalability and sparsity issues in recommender datasets: a survey. *Knowledge and Information Systems* 62, 1 (2020), 1–43.

[16] Xiaoyuan Su and Taghi M Khoshgoftaar. 2006. Collaborative filtering for multi-class data using belief nets algorithms. In *2006 18th IEEE international conference on Tools with Artificial Intelligence (ICTAI'06)*. IEEE, 497–504.

[17] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Neural Information Processing Systems Conference (NIPS 2013)*, Vol. 26. Neural Information Processing Systems Foundation (NIPS).

[18] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems.. In *NIPS*. 4957–4966.

[19] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 448–456.

[20] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1235–1244.

[21] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[22] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[23] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *IJCAI*, Vol. 17. Melbourne, Australia, 3203–3209.

[24] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[25] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. *arXiv preprint arXiv:1905.13129*.

[26] Zi-Ke Zhang, Chuang Liu, Yi-Cheng Zhang, and Tao Zhou. 2010. Solving the cold-start problem in recommender systems with social tags. *EPL (Europhysics Letters)* 92, 2 (Oct. 2010), 28002. https://doi.org/10.1209/0295-5075/92/28002 arXiv:1004.3732 [cs.IR]