CONDITIONAL PREFERENCE NETS FOR USER AND ITEM COLD START PROBLEMS IN MUSIC RECOMMENDATION

Szu-Yu Chou[1,2], Li-Chia Yang[2], Yi-Hsuan Yang[2], Jyh-Shing Roger Jang [1]

[1]Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan
[2]Research Center for IT innovation, Academia Sinica, Taipei, Taiwan
{fearofchou, richard40148, yang}@citi.sinica.edu.tw, jang@csie.ntu.edu.tw

ABSTRACT

A great amount of data is usually needed for a recommender system to learn the associations between users and items. However, in practical applications, new users and new items emerge everyday, and the system has to react to them promptly. The ability to recommend proper items to new users affects the users' first impression and accordingly the retention rate, whereas recommending new items to proper users contributes to the freshness of the recommendation. In this paper, we propose a deep learning model called the conditional preference nets (CPN) to deal with both new users and items under the same model framework. CPN employs an introductory user survey to learn about new users, and content features automatically extracted from items for the item side. Through a new idea called the preference vectors and an existing content embedding technique, the same model can capitalize the observed associations between known (i.e. old) users and items, thereby benefiting from the cumulative knowledge of user behavior gained over time. We validate the superiority of CPN over prior arts using the Million Song Dataset. We also demonstrate how CPN allows a user to pick either genres, artists, or the combination of them in the introductory survey.

Index Terms— Content-based recommendation, cold start problem, introductory survey, convolutional neural network

1. INTRODUCTION

Recommendation has been considered as a core component in many online services such as Netflix or Amazon [1]. Most recommender systems model user preference by mining massive user behavior data, which manifests in either explicit user ratings or implicit feedback such as item clicks and play counts [2, 3]. The more we know about the users and items, usually the easier we can generate good recommendation.

There are two primary approaches to recommendation: collaborative filtering (CF) [2, 4, 5] and content-based filtering (CB) [6–9]. The CF approach has been shown effective in various domains, including movie, music, and news [10]. However, as CF identifies like-minded users solely from the user behavior data, it cannot work well for users or items with little or no past information available, such as newly released or unpopular items. The CB approach tackles this *cold start* problem by using additional content features extracted from the items, for example to select new items that are similar to existing items that a user likes. However, most existing CB methods can deal with only the *item cold start* problem [11–13] rather than the *user cold start* problem, due to the difficulty of acquiring personal information from, and in general finding a suitable feature representation for users [14, 15].

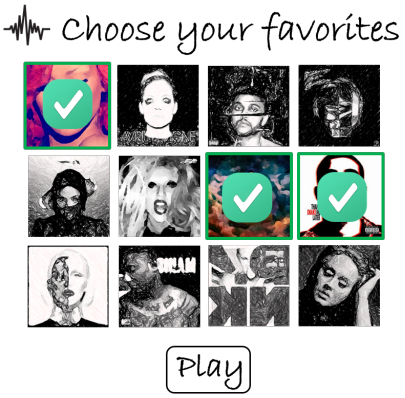

Fig. 1: Illustration of an introductory survey for people using an online music service for the first time. The users are asked to pick their favorite artists from a pre-defined list of candidates, from which a preference model is learned for each of them.

The user cold start problem emerges as there are new users joining an online service everyday [15]. While the item cold start problem is related to the coverage and freshness of recommendation, the user cold start problem affects the new users' first impression of the system and accordingly their conversion or retention rate, which is critical to a commercial system. A solution commonly adopted by the industry is to probe a new user's preference through an inexpensive, short *introductory survey* [14], by for example asking the users to propose or to select their favorite items, as illustrated in Figure 1. Some heuristics can then be employed to recommend items to these users, for example by suggesting the items that are content-

wise most similar to the self-report favorite ones [14].

What we are interested in is an end-to-end machine learning model that can deal with both the user cold start and item cold start problems at the same time, a task that is seldom addressed in the literature. In particular, being motivated by the success of deep learning models in pattern recognition problems [16], we choose to investigate deep learning models in this paper, using a music dataset of tens of thousands of users and songs to train and validate our model.

In view of the convenience and low cost of the introductory survey, the goal is to build a hybrid CF/CB deep learning model that can exploit all the following three information sources: 1) observed behavior data for known (i.e. old) users and items, 2) result of the introductory survey of new users, and 3) content features for (both old and new) items. We find that the action of selecting items in the introductory survey can be viewed as constructing a *preference vector* on which the neural network is *conditioned* [17, 18]. In consequence, we call the model the *conditional preference nets*.

Specifically, as the input for the user, we propose a feature representation called the preference vector that can be constructed from either the user behavior data or the introductory survey on-the-fly. For the item side, adapting the idea of content embedding [11], we set the learning target by the behavior data of existing users and learn layers of parameters in the offline training phase to process both the raw item content features and the user preference vectors. We note that while the existing content embedding method learns features only for items [11], ours is for both users and items.

The main contribution of the paper is twofold:

- We propose a flexible preference vector to be used as the feature input for the user side in a deep neural network. We can compute the preference vector for a new user through an online introductory survey, thereby alleviating the user cold start problem in recommendation.
- We propose a novel end-to-end deep learning model to learn user preference through behavior data and item content features, conditioned on user-specific preference vectors. As the model is content-based, it can recommend unpopular or new items, providing a solution to the item cold start problem.

In what follows, we present the proposed model in Section 2, the experimental setup and dataset in Section 3, and finally the experimental results in Section 4.

2. METHODOLOGY

2.1. Preference Vector

A preference vector $\mathbf{x}_u = [r_{u1}, r_{u2}, \ldots, r_{u|\mathcal{I}|}]^T$ specifies the items that a user u likes, where $|\mathcal{I}|$ denotes the number of elements in the item list $\mathcal{I}$ and $r_{ui} > 0$ if the association exists. It can be constructed for a new user on-the-fly through an introductory survey. A user may want to select their favorite items (e.g. from a pre-defined list) without giving any weights, leading to a binary preference vector, i.e. $r_{ui} \in \{0, 1\}$. Or, a user may want to explicitly rate his or her preference of the items. A preference vector of the same length can also be built for an old user from the user behavior data, by checking whether or how much the user likes an item in the list. Therefore, the preference vectors for old and new users are comparable.

Although being simple, the preference vector can be considered as the *raw* data encoding a user's preference, comparing to other feature representations one might compute to characterize the user. However, for computational reasons, we might want to avoid using all the individual items (e.g. songs) in a database for the preference vector, when it leads to exceedingly large $|\mathcal{I}|$. For many applications we can use domain knowledge to reduce $|\mathcal{I}|$ by using item groups (e.g. albums, artists or genres) instead. For recommendation applications, this has the additional advantage of reducing the cognitive load of the introductory survey, as the candidate list is shorter.

In a real-world application, the items in the database may be dynamic: items are added to and removed from the database everyday. Moreover, we might want to update the candidate list in the introductory survey over time to account for changes in the mainstream preferences. These engineering issues are out of the scope of this paper, but they can be addressed with some adaptations of the preference vector, such as using different subsets of $\mathcal{I}$ over time as the candidate list for the survey.

2.2. Conditional Preference Nets (CPN)

Assume we are given the user behavior data $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{J}|}$ for known users $\mathcal{U}$ and items $\mathcal{J}$, where $\mathcal{J}$ contains all the individual items in the database (usually $\mathcal{J} \supseteq \mathcal{I}$). We are also given the preference vectors for these users, $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$, and the *raw* content features $\mathbf{\Psi} \in \mathbb{R}^{|\mathcal{J}|\times m}$ for the items, where m denotes the dimensionality of the content features. The task is to learn a model that maximizes the joint probability:

$$P(\mathbf{R} \mid \mathbf{X}, \mathbf{\Psi}) = \prod_{(u,j)\in\mathbf{R}} P(d_{uj} \mid \mathbf{x}_u, \psi_j)^{c_{uj}}\,, \qquad (1)$$

where d_{uj} is a binarized version of r_{uj} indicating whether such a user-item association exists, and c_{uj} is the *confidence* of the association parameterized by two parameters α and ε:

$$c_{uj} = 1 + \alpha \log(1 + r_{uj}/\varepsilon)\,. \qquad (2)$$

That is to say, we use the binary-valued d_{uj} as the training target and the confidence scores c_{uj} to weigh different data instances. Similar to [19], in our pilot study we found that such a non-linear scale of r_{uj} performs remarkably better than a linear scale in reducing the bias from heavy users.

Our model fits the behavior distribution by mapping learned representation $\mathbf{x}_u^l$ and ψ_j^l into a joint space:

$$\mathbf{z}_{uj} = f(\mathbf{W}_x \mathbf{x}_u^l + b_x^l) \circ f(\mathbf{W}_\psi \psi_j^l + b_\psi^l)\,, \qquad (3)$$

where $\mathbf{x}_u^l$ and ψ_j^l is the result from $f(\mathbf{W}_x\mathbf{x}_u^{l-1} + b_x^{l-1})$ and $f(\mathbf{W}_\psi * \psi_j^{l-1} + b_\psi^{l-1})$ respectively, $*$ denotes the convolution operator, $f(\cdot)$ the rectified linear activation function, $\mathbf{W}$ the parameters, and b the bias term. The variable l indexes the layers in the deep neural network, and when $l = 1$ we have $\mathbf{x}_u^1 \triangleq \mathbf{x}_u$ and $\psi_j^1 \triangleq \psi_j$. In Eq. (3) we combine information from the user and items sides by the element-wise product, denoted as $\circ$. An alternative is to simply concatenate the two outputs, but in our experiment we found that the element-wise product works slightly better. A theoretical justification of doing so is to enforce the neural network to map user preferences and item content features to a joint space.

Finally, the prediction $\hat{d}_{uj}$ can be computed by using specific $\mathbf{z}_{uj}$ from user u and item j. The prediction is defined as $\hat{d}_{uj} = \sigma(\mathbf{V}^{\mathrm{T}}\mathbf{z}_{uj}^l)$, where $\sigma(\cdot)$ is the sigmoid function and $\mathbf{z}_{uj}^l$ is the result from $f(\mathbf{W}_z\mathbf{z}_{uj}^{l-1} + b_z^{l-1})$.

Training: The learning objective of CPN is to predict the user-item association d_{uj}. We use the following cost function:

$$\min_{\theta} -\frac{1}{|N|}\sum_{u,j} c_{uj}(d_{uj}\log\hat{d}_{uj} + (1-d_{uj})\log(1-\hat{d}_{uj}))\,, \quad (4)$$

where θ contains all the parameters of the network. As it is important to have negative data in training a model, we randomly sample user-item pairs with no association (i.e. $d_{uj} = 0$) for *negative sampling*, ensuring that each user and item in the training set would be at least sampled once. The model parameters θ are optimized by using stochastic gradient descent (SGD), which computes the gradients using a backpropagation algorithm. Specially, we employ the subgradient method AdaGrad [19] to adaptively set the learning rate for speeding up the convergence rate. To prevent overfitting, we also randomly drop out some neurons while training.

Implementation details: We use the time-frequency representation mel-spectrogram as the raw content features of music, for it is commonly used in deep learning models dealing with audio signals [11,20]. We compute it using short-time Fourier transform with 4,096-sample, half-overlapping windows, for musical audio sampled at 22kHz. The mel-scale is used to reduce the dimensionality along the frequency axis to 128. The convolution network for the item side contains two convolutional layers: the first layer has 256 convolutional filters of size 128×4 and max pooling kernel 1×4, and the second layer has 512 convolutional filters of size 1×4 and max pooling kernel 1×2. The convolution is applied only along the time dimension. After the convolution layers, the representation is further optimized by fully-connected layers with 2,048 neurons. For the preference vector on the user side, we use two fully-connected layers with 2,048 neurons. After the joint layer that combines the neural network outputs from the user and item sides, we use another two fully connected layers with 1,024 neurons to predict the user-item association.

2.3. Offline Simulation Framework for New Users

We need an online introductory survey to build the preference vectors for new users in the cold start setting. Following Rashid *et al.* [15], we use an offline simulation framework that assumes a new user will always pick the items that the user likes the most (e.g. the songs with the highest playcounts) in the survey. As a user is unlikely to pick too many items in the survey, in our implementation we assume that we are given only the top 5 favorite items of a user from the survey.

To reduce the size of $\mathcal{I}$, we use the artists and genres presenting in the database as the item groups. Specifically, this is done by concatenating a binary vector of artists and a binary vector of genres, where we have 1's for the artists corresponding to and the genre tags associated with the 5 favorite songs. With this setting, we also like to demonstrate the flexibility of the preference vector: we can use various types of item groups in the introductory survey, as long as there is a mapping between the elements $\mathcal{I}$ in the preference vector and the set of individual items $\mathcal{J}$. Moreover, although not demonstrated in this paper, thanks to the flexibility of the preference vector, the recommendation model can be conditioned on any other feature descriptors of users (e.g. age) by adding these features to the preference vector, as long as such information is available for both the training and test users.

3. EXPERIMENTAL SETUP

Dataset: The Million Song Dataset (MSD) [21] is a public dataset comprising of audio features and metadata for about a million contemporary songs. We use its taste profile subset in our experiment. It contains the playcounts of over a million users for over 380,000 songs. We keep the songs whose 30-second audio previews can be crawled from the 7digital website (so as to compute the mel-spectrograms from audio) and those whose genre labels are available in the metadata (to construct the preference vector and to interpret the recommendation result). Moreover, we randomly pick users who have listened to more than 10 songs among these songs. The dataset is then split into a training subset, a validation subset, and two test subsets. The two test subsets are used to evaluate the recommendation result in the warm start (to recommend known items to known users) and cold start (to recommend *new* items to *new* users) settings, respectively. For example, in the cold start setting, the task is to rank the new items according to how likely a new user would like them. See Table 1 for the statistics of the data subsets. There are 20,473 artists and 60 genres in total. The final dataset is still a big one, permitting the use of a deep neural network for learning.

Evaluation metric: A typical way to evaluate the performance of recommendation is to assess whether any of the top N items recommended to a user are favored by the user or not [22]. The average result for all the test users is reported, using the following three standard metrics: recall, precision

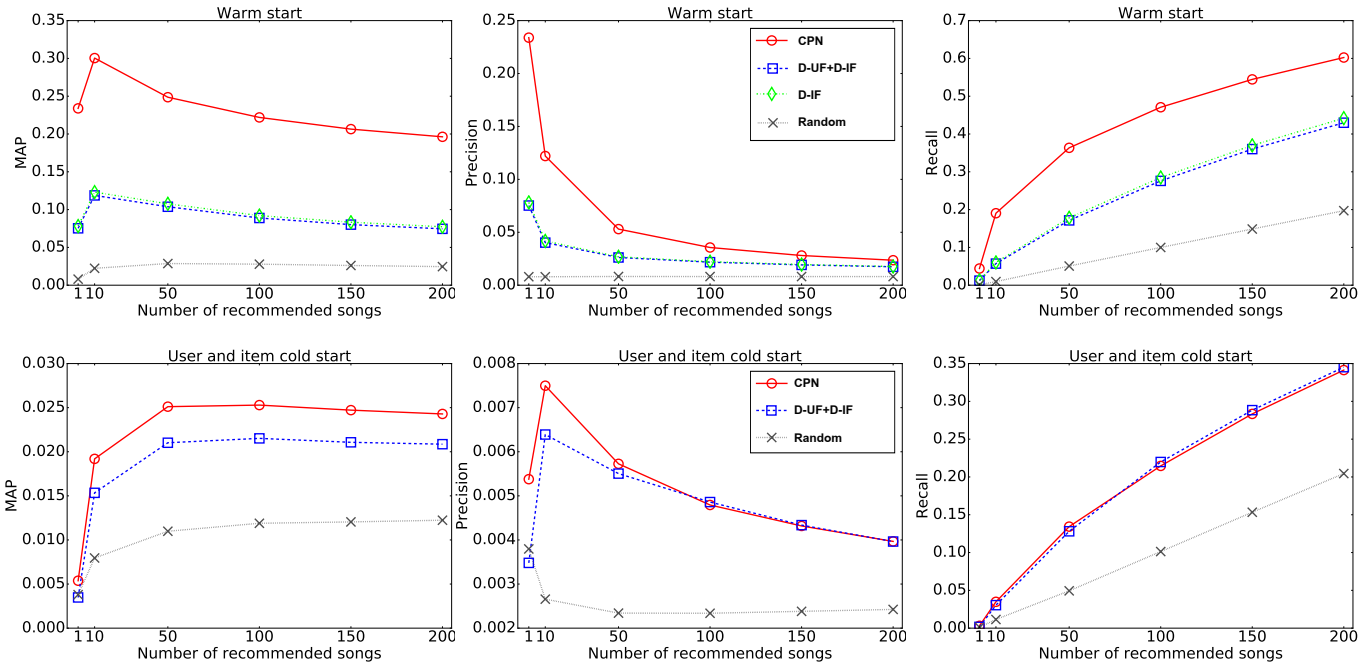

Fig. 2: Performance of different methods in the (upper) warm start and (lower) cold start settings, evaluated in terms of (from left to right) MAP, precision and recall at different ranks (i.e. from 1, 10, 50, 100, 150 to 200) of the recommended items.

Table 1: Statistics of different subsets of the dataset

Data sets	#users	#songs	#user-song pairs
Training	25,000	112,908	763,929
Validation	22,075	34,373	576,125
Test (warm start)	24,344	50,305	202,847
Test (cold start)	3,161	2,209	7,553

and mean average precision (MAP). Moreover, following Cremonesi *et al.* [22], to avoid overestimating the accuracy of the recommendation result, we randomly select 1,000 additional non-listened items for each user (i.e. $d_{uj} = 0$) as the negative data and add them to the pool of candidate songs for ranking.

Baselines: We compare the performance of CPN with an existing deep learning model and a variant that we propose.

- **D-IF** [11] is a state-of-the-art CB method that uses a deep learning model to deal with the item cold start problem. This method contains two steps. First, it uses weighted matrix factorization (WMF) [2], a state-of-the-art CF method, to learn fixed-dimensional *latent factors* of users and items from the user behavior data. Then, a CNN model is trained by treating the item latent factors as the learning target and item content features such as the mel-spectrogram for music as the input. The inner product of the user latent factors learned by WMF and the *item feature embeddings* learned by the CNN is then used to estimate how likely a user likes an item.
- **D-IF+D-UF**: To deal with both user cold start and item cold start, following the idea of D-IF, we train an additional deep neural network that uses the user latent factors computed by WMF as the learning target and the proposed user preference vectors as the input. For recommendation, we compute the inner product of the item feature embeddings and the *user feature embeddings*.

Although many recommendation algorithms have been proposed in the literature, we consider mainly D-IF and its variant, for D-IF represents the state-of-the-art CB method for music recommendation. Many other methods, such as WMF, cannot deal with either the user cold start or item cold start problem. Whenever possible, we use similar parameter settings and inputs for CPN and these methods for fair comparison.

4. EXPERIMENTAL RESULTS

Figure 2 shows the result of different methods for the warm start and cold start settings, including the result of a random baseline that randomly ranks the songs. We see similar trends from the result of the considered methods: the MAP and precision saturates to certain values as the number of recommended items is sufficiently large, while the recall keeps going up along with the number of recommended items. All the deep learning methods perform remarkably better than the random baseline. For the warm start setting, we see that the proposed CPN model performs remarkably better than the competing methods, suggesting the advantage of learning to predict directly

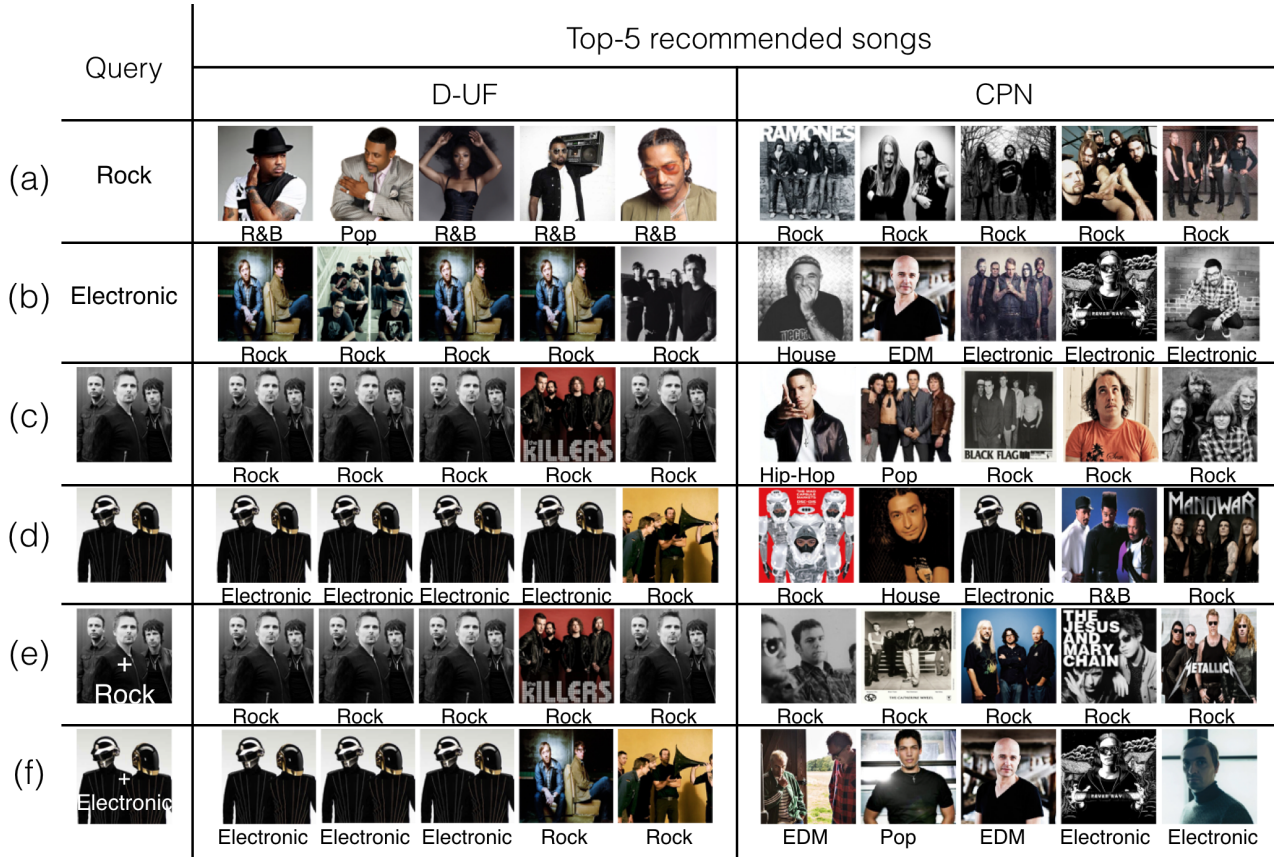

Fig. 3: The top-5 recommended songs for an imaginary user who selects different item groups (genre, artist, or a pair of genre and artist) in the introductory survey. Note that CPN employs item content features, but D-UF does not. The recommended songs are visually presented by an image of the corresponding artist and the genre tag provided by MSD.

the user-item association in an end-to-end manner. In contrast, both D-IF and D-UF+D-IF employ a two-stage approach that predicts the latent factors computed by WMF instead of the user-item association. For the cold start setting, the values in all the three performance metrics are much lower than those in the warm start setting, suggesting the difficulty of recommending new items to new users. However, we see that CPN can still outperform D-IF+D-UF in MAP and precision.

Next, for a qualitative evaluation, we employ again the offline simulation framework and show the recommendation result when an imaginary user selects different genres, artists, or the combination of them in the introductory survey. This can give us some insights into how the recommender system responses to different inputs from the introductory survey.

To demonstrate the importance of taking into account item content features, we compare the result of CPN with D-UF:

- **D-UF**: Similar to D-IF+D-UF, we train a deep neural network to learn the mapping between user preference vectors and user latent factors computed by WMF. However, for recommendation, we use the inner product of the user feature embeddings learned by the neural network and the item latent factors computed by WMF, without using the item content features at all. Unlike D-IF, D-UF can deal with the user cold start problem.

Figures 3a and 3b show the top-5 songs recommended by D-UF and CPN when a user picks different genres in the introductory survey. When the user chooses *Rock*, we see that CPN also recommends Rock songs in return. On the other hand, when the user chooses *Electronic*, CPN recommends not only Electronic music but also House and Dance music, which are indeed related to Electronic.

Figures 3c and 3d show the recommended songs when the user selects only one particular artist. We see that CPN can indeed recommend songs of related genres, and the songs are from different artists. In contrast, although the recommendation of D-UF is also relevant, the recommended songs lack diversity as they are mostly from the specified artist.

Finally, Figures 3e and 3f show the recommended songs when the user picks a genre and an artist. We can see that the recommendation result is different from the case where only a genre or an artist is selected, and that the result of CPN is more diverse, suggesting the benefit of using item content features.

5. CONCLUSION

In this paper, we have presented a new deep learning model called conditional preference nets (CPN) to jointly optimize the user and item features for predicting the user-item association in an end-to-end manner. To accommodate users new to a recommender system, we have also presented the idea of preference vectors to condition the neural network. Because CPN can jointly leverage user behavior data, item content features, and inputs from the introductory survey, it can deal with both the user cold start and item cold start problems. Our experiment on a music dataset shows that CPN provides relevant and diverse recommendation. As CPN is a generic method, it can also be applied to any other domains.

For future work, we are interested in combining the idea of active learning [23] to CPN through manipulating the preference vectors, in order to model the preference of a new user with fewer inputs. We also want to study other item picking strategies in our offline simulation framework. By modifying the input and structure of the neural network, it is also possible to study other applications such as next-item recommendation [24] or cross-domain recommendation [25].

6. REFERENCES

[1] J. Bennett and S. Lanning, "The Netflix Prize," in *Proc. KDD Cup Workshop*, 2007.

[2] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. IEEE ICDM*, 2008, pp. 263–272.

[3] D. Parra and X. Amatriain, "Walk the talk: Analyzing the relation between implicit and explicit feedback for preference elicitation," in *Proc. Int. Conf. User Modeling, Adaption, and Personalization*, 2011, pp. 255–268.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. WWW*, 2001, pp. 285–295.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[6] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Proc. National Conf. Artificial Intelligence*, 2002, pp. 187–192.

[7] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. ACM SIGIR*, 2002, pp. 253–260.

[8] P. Forbes and M. Zhu, "Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation," in *Proc. ACM Recsys*, 2011, pp. 261–264.

[9] S. Rendle, "Factorization machines with libFM," *ACM Trans. Intelligent Systems and Technology*, vol. 3, no. 3, pp. 57:1–57:22, May 2012.

[10] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. in Artif. Intell.*, pp. 1–19, 2009.

[11] A. v. d. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. NIPS*, pp. 2643–2651. 2013.

[12] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proc. ACM Multimedia*, 2014, pp. 627–636.

[13] S.-Y. Chou, Y.-H. Yang, J.-S. Jang, and Y.-C. Lin, "Addressing cold start for next-song recommendation," in *Proc. ACM RecSys*, 2016, pp. 115–118.

[14] D. Kluver and J. A. Konstan, "Evaluating recommender behavior for new users," in *Proc. ACM RecSys*, 2014, pp. 121–128.

[15] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: An information theoretic approach," *Proc. ACM SIGKDD Explor. Newsl.*, pp. 90–100, 2008.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[17] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[18] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelCNN decoders," *CoRR*, vol. abs/1606.05328, 2016.

[19] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proc. ACM SIGIR*, 2015, pp. 373–382.

[20] J.-Y. Liu and Y.-H. Yang, "Event localization in music auto-tagging," in *Proc. ACM Multimedia*, 2016, pp. 1048–1057.

[21] B. Mcfee, T. Bertin-Mahieux, D. P. W. Ellis, and G. R. G. Lanckriet, "The Million Song Dataset Challenge," in *Proc. AdMIRe*, 2012.

[22] P. Cremonesi, Y. Koren, and P. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proc. ACM Recsys*, 2010, pp. 39–46.

[23] A. S. Harpale and Y. Yang, "Personalized active learning for collaborative filtering," in *Proc. ACM SIGIR*, 2008, pp. 91–98.

[24] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proc. WWW*, 2010, pp. 811–820.

[25] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proc. WWW*, 2015, pp. 278–288.