Categorical-Attributes-Based Item Classification for Recommender Systems

Qian Zhao*
University of Minnesota
Minneapolis, Minnesota, United States
zhaox331@umn.edu

Jilin Chen, Minmin Chen, Sagar Jain
Alex Beutel, Francois Belletti, Ed H. Chi
Google Inc.
Mountain View, California, United States
{jilinc,minminc,sagarj,alexbeutel,belletti,edchi}@google.com

ABSTRACT

Many techniques to utilize side information of users and/or items as *inputs* to recommenders to improve recommendation, especially on cold-start items/users, have been developed over the years. In this work, we test the approach of utilizing item side information, specifically categorical attributes, in the *output* of recommendation models either through multi-task learning or hierarchical classification. We first demonstrate the efficacy of these approaches for both matrix factorization and neural networks with a medium-size real-word data set. We then show that they improve a neural-network based production model in an industrial-scale recommender system. We demonstrate the robustness of the hierarchical classification approach by introducing noise in building the hierarchy. Lastly, we investigate the generalizability of hierarchical classification on a simulated dataset by building two user models in which we can fully control the generative process of user-item interactions.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

recommender systems; hierarchical softmax; hierarchical classification; multi-task learning

ACM Reference Format:
Qian Zhao and Jilin Chen, Minmin Chen, Sagar Jain Alex Beutel, Francois Belletti, Ed H. Chi. 2018. Categorical-Attributes-Based Item Classification for Recommender Systems. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240367

1 INTRODUCTION

Matrix factorization and deep neural networks, such as Recurrent Neural Networks[12] (RNN), trained on user-item co-occurrence data have become the default modeling choice for recommendation systems. Nevertheless, the extreme sparsity of the co-occurrence data still poses a huge challenge. Various algorithms have been proposed to go beyond user-item co-occurrence data and take into account item (or user) side information, *i.e.* the attributes of the items or users. In particular, categorical attributes or taxonomies of items offer similarity measurement between items, which is critical for recommendation when the user-item interaction pair is sparse. SVDFeature [7] and Factorization Machine [20] are a few successful examples. In these approaches, item attributes are embedded in the *input* and modeled together with the user-item co-occurrence data.

The structure of the item space induced by these categorical attributes are often not utilized on the *output* side of the models though. In this work, we propose to introduce auxiliary task to predict item categorical attributes with the goal of improving the task of item recommendation (prediction). Intuitively, more observations are available on these coarser-granularity categorical attributes than individual item. As a result, predicting these attributes is easier than predicting the item directly. The structural relationship between the item attributes and the item itself, in return allows us to utilize the auxiliary task to help item recommendation. Here we focus on a particular setting where recommendation is modeled as a multi-class classification problem, in which the number of classes is the number of unique items in the system. We employ two modeling techniques to introduce the auxiliary task of predicting item attributes: hierarchical classification or hierarchical softmax [18] (*HSM*) and multi-task learning [21] (*MTL*).

HSM is not a new technique but remain competitive and popular. It is commonly used to speed up learning with a large number of output classes. Morin and Bengio showcased its efficiency in the context of language modeling[18], especially when incorporating the hierarchy of words inherited from WordNet. Similarly, multi-task learning has been explored extensively in prior work, where shared model parameters between related prediction tasks can help the main prediction task through regularization and transfer learning [3, 21].

Predicting coarser concepts like people, topics (or tags) are not new in recommender systems either, *e.g.*, the tag recommender [22], followee recommendation [6], etc. However, these work aims at predicting these coarser attributes as the end goal. In contrast, we are more interested in improving the item-level prediction by introducing these coarser level targets into the modeling.

We demonstrate the effectiveness of the proposed approach in modeling frameworks of both matrix factorization and neural networks. We compare the performance of proposed approaches against several baseline methods on utilizing item side information on a public real-world dataset as well as a private industrial-scale

*Work performed while at Google.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10.
https://doi.org/10.1145/3240323.3240367

recommendation dataset. While both MTL and HSM offer noticeable improvement over baselines, we find that HSM boosts the performance substantially and consistently. We also examine the robustness of HSM w.r.t. noise in the group assignment based on the attributes. As expected, the improvement drops as more noise are injected into the hierarchy. Nonetheless, it achieves comparable performance to baseline using a completely random hierarchy. Through simulation, we also demonstrate that HSM's advantage generalizes across different user models where user-item interaction follows different generative models.

2 RELATED WORK

2.1 Matrix Factorization and Neural Networks in Recommender Systems

Historically, the matrix completion problem has been extensively studied in the algorithmic research of recommender systems. The techniques developed can model both user explicit (*e.g.* five-star preference ratings) or implicit feedback (*e.g.* click, purchase, watch) data. Nowadays, more and more researchers focus on designing models for broader user implicit feedback especially in industrial recommender systems where users' natural engagement with products provide a large amount of useful information on users' interest. For instance, Hu et al. [14] proposed collaborative filtering models for implicit feedback data, utilizing a efficient negative sampling technique to address the modeling challenge that only implicit positive labels are observed in these recommender systems, *i.e.*, only items interacted with by users are observed while user preference information on unobserved items are missing. Negative sampling technique has been widely used and demonstrated effective in many applications, e.g. K. Chen and T. Chen et al.'s work on tweet and followee recommendation [5, 6].

To deal with cold-starting items and users, the state-of-the-art generalized matrix factorization models, *e.g.* SVDFeature [7] and Factorization Machine [20] can easily incorporate item side information as input. Compared with user ID or item ID alone, utilizing these more higher-level user or item features are important for generalization when there are not many observations for the user or item.

Recently, deep learning has shown great accuracy improvements in various domain tasks, including recommender systems where temporally long-term dependencies might be important for modeling user behavior. Wu et al.'s recent work [25] models users and movies with a LSTM autoregressive model in movie recommendation systems and offers better prediction accuracy than the traditional matrix factorization techniques. LSTM is a widely used recurrent neural networks due to its advantage of modeling long-term dependencies.

2.2 Multi-Task Learning

Multi-task learning (MTL) has led to successes in many applications of machine learning. Ruder [21] gave a comprehensive overview on MTL particularly in deep neutral networks. They discussed different auxiliary tasks that can be used to leverage MTL when even only one task is the target. They pointed out that in the current status of the field, finding an auxiliary task is largely based on the assumption that the auxiliary task should be related to the main task in some way and the field still does not have a good notion of when two tasks should be considered similar or related.

One quite similar finding to our work is in Caruana et al's work [4], which showed that some features are more useful as extra outputs than as inputs. They demonstrated through two regression problems and one classification problem that using some features as an output can enable learning a mapping from the other inputs to that feature which turns out to be more useful that the feature values themselves provided as input. In our work, we present similar findings but with a special focus on recommender systems and item categorical features. Different from co-factorization proposed by Singh *et al.* [23] where the matrix of item-attribute co-occurrences was fitted together, we additionally model a derived user-attribute matrix where the attributes come from items that users interacted with.

2.3 Hierarchical Prediction

As mentioned previously, hierarchical softmax was proposed by Morin and Bengio in [18] where it is used on top of a binary word hierarchy with multiple levels. They demonstrated that theoretically hierarchical softmax can have exponential speed-up in both training and testing (if only the top-1 prediction is needed during testing). If predictions on all items in the vocabulary are needed, there is a constant factor overhead (which is the case in recommender systems). In the work, they also proposed to share parameters across the hierarchy which we adopted in our work (see the next section). Empirically, they show that a hierarchical decomposition of the softmax yields a speed-up of about 200 both during training and inference. The limitation of this work is that the hierarchy was manually curated through data-driven clustering process. Mnih and Hinton [17] further proposed a simple and fast feature-based algorithm for automatic construction of such hierarchies.

We applied the hierarchical softmax model in recommender systems based on hierarchies induced by the naturally available item categorical attributes. In addition, we made changes on top of the model for tractability issues when the model has a huge flat hierarchy instead of the binary deep hierarchy used in [18] and when the hierarchy is very unbalanced which is often the case for these categorical attributes. Interestingly, Covington *et al.* [8] briefly mentioned their work of using hierarchical softmax model for Youtube recommendation, which did not achieve comparable results compared with random negative samples. However, we recognize that the clusters used there seem to be randomly assigned (which is confirmed in one of our experiments) because they mentioned that traversing each node in the hierarchical tree involves discriminating between sets of classes that are often unrelated.

Hierarchical classification with decision trees has also been used in genomics. Vens et al. [24] proposed hierarchical multi-label classification (HMC) trees for hierarchical multi-label classification tasks. Compared with two baseline approaches: hierarchical single-label classification (HSC) and single-label classification (SC) trees, HMC trees outperform along three dimensions: predictive accuracy, model size and induction time. The hierarchy of classes can be a tree where each class has only one parent, or a Directed Acyclic Graph (DAG) where classes may have multiple parents. They show how HMC trees can be extended to support this setting. In our

work, we studied the case where only one categorical attribute is selected for the item hierarchy and each item can only belong to one categorical attribute value, *i.e.*, one parent in the hierarchical structure. We treat it as interesting future work to support modeling multiple-parents or multi-attribute hierarchical classification for recommender systems.

3 UTILIZING SIDE INFORMATION IN RECOMMENDER SYSTEMS

Recommendation modeling involves modeling both users and the item prediction. Most of the techniques model the current user state and items with a low-dimensional vector space and predict the user's preferences on items by matching the two. Assume we have a user state model that combines user profile and context factors and lies in a low-dimensional space R_d. Denote $s = s(u) \in R_d$ as the current user's state vector in which u represents the user ID, profile and history in the system. We also sometimes use notation u to refer to a specific user ID depending on the context. Denote $o = o(v) \in R_d$ as the item representation vector where v represents the item ID a and the properties of an item including the item side information. Denote θ as the model parameters of both s and o.

With the user state s and item representation o, making predictions on items involves modeling the match between the two $f(u, o)$. This prediction function is used to approximate or learn from user feedback signals, which could be user explicit ratings on items or user implicit actions (*e.g.* interaction, consumption etc.) on items. Here we focus on the case of implicit action feedback. In most recommender systems, this implies that the system only observes positive feedback missing negative observations, *e.g.*, we observe a user consumes an item but we miss information on the user's preference on other unobserved items. Negative sampling is widely used in this implicit feedback case, *i.e.*, randomly sample some items in the item space as the "pseudo-negative" observations.

Three possible ways of fitting the observations can be found in prior literature: *logistic model* (treating the observations as following independent Bernoulli distributions), *pair-wise ranking model* (modeling the relative preference order) and *softmax model* (treating the feedback as an observation of an exclusive multi-class classification model). In this work, we focus on the softmax model because on one hand one of our proposed approaches – hierarchical softmax – can only be applied in this case and on the other hand softmax model has been demonstrated to have benefits [26]. The softmax model is shown in Equation 1, where i are k are indexing in the $\alpha + 1$ observations, which include one positive and other sampled negative ones, *i.e.*, α is the negative sample size and a here is an one-hot encoded representation of the positive and negative item IDs.

$$\widehat{p_a}(a_i = 1) = \frac{exp(f(s, o_i))}{\sum_{k=1}^{\alpha+1} exp(f(s, o_k))} \tag{1}$$

3.1 Matrix Factorization: SVDFeature

Many matrix factorization based techniques have been developed to utilize user and item side information to make predictions in recommender systems, *e.g.*, Factorization Machine and SVDFeature. In this work, without losing generality, we examine the SVDFeature model. The SVDFeature model has the following prediction function shown in Equation 2, 3, 4 (the bias terms are skipped for simplicity of presentation). In other words, SVDFeature represents the user state and item vector as the sum of the vectors of their ID and side information. In the simplest case where users are represented as simple embedding vectors U and items as $W \in R_{Nxd}$ which have a categorical attribute represented as $V \in R_{Mxd}$, s becomes $s(u) = U_u$ and o becomes $o(v) = W_a + V_c$ where a represents the item ID and c represents a's categorical attribute ID.

$$f(u, o) = s^T o \tag{2}$$

$$s(u) = \sum_i u_i \cdot \theta_{u_i} = U_u \tag{3}$$

$$o(v) = \sum_j v_j \cdot \theta_{v_j} = W_a + V_c \tag{4}$$

3.2 Neural Networks: RNN

Neural network based techniques can incorporate side information as well, *e.g.*, by embedding not only the user or item ID but also their side information into a low-dimensional space and then concatenating the vectors to have a unified vector representation as the input of a neural network. Without losing generality, we examined a Recurrent-Neural Network (RNN) based model that models a user as the user's temporal interaction sequence in the system [12, 25]. For a user sequence, each step is a vector representation of the item that the user interacted with (or liked, consumed), concatenating the embeddings of not only the item ID (*input-side* embedding of an item) but also its side information, overall denoted as $CAT(\theta_{u_t})$. One widely used step transition model for the sequence model is LSTM [13]. Additional layers of transformation can be applied before modeling the transition with LSTM, *e.g.*, through a layer of Rectified Linear Units (ReLU) [19]. In summary (as illustrated in Figure 1), a RNN model has the following user state model s in Equation 5 where t represents the time step.

$$s_t(u) = LSTM(s_{t-1}(u), ReLU(CAT(\theta_{u_t}))) \tag{5}$$

In order to make predictions, the RNN model introduces additional *output-side* embedding vectors for items W. Therefore, the prediction function of the RNN model becomes Equation 6:

$$f(u, o) = s^T W_a \tag{6}$$

4 PREDICTING ITEM CATEGORICAL ATTRIBUTES

In this work, we propose to enhance the previous recommendation models by further utilizing the item side information in the output part. Specifically, we introduce an auxiliary task making predictions on not only the item (*i.e.*, the match $f(u, a)$) but also on one of the item's categorical attributes. We only consider utilizing one categorical attribute at a time and the case that there is only one value associated with the attribute. In cases where there are multiple values for the attribute, we pick one according to the criterion of

Table 1: Results for the Behance data set: The testing MAP@k (k=5 or 20) and AUC of the item and attribute prediction tasks for MTL and HSM on top of both SVDFeature and RNN with the owner as the auxiliary task. The numbers in the parentheses are the MAP improvement relative to the baseline model SVDFeature or RNN. Best item MAP and AUC are bolded.

Model	Item MAP@5	Item MAP@20	Item AUC	Attribute MAP@5	Attribute MAP@20	Attribute AUC
SVDFeature	0.0035	0.0046	0.759	N.A.	N.A.	N.A.
SVDFeature+MTL	0.0044 (+25.7%)	0.0057 (+23.9%)	0.758 (-1.3%)	0.020	0.023	0.845
SVDFeature+HSM	**0.0046 (+31.4%)**	**0.0066 (+43.4%)**	**0.813 (+7.1%)**	0.025	0.029	0.847
RNN	0.0099	0.0121	0.758	N.A.	N.A.	N.A.
RNN+MTL	0.0104 (+5%)	0.0129 (+6.6%)	0.759 (+1.3%)	0.027	0.031	0.839
RNN+HSM	**0.0129 (+30.3%)**	**0.0159 (+31.4%)**	**0.787 (+3.8%)**	0.024	0.028	0.842

Table 2: Results for the Behance data set: The testing MAP@k (k=5 or 20) and AUC of the item and attribute prediction tasks for SVDFeature and SVDFeature+HSM whose group assignment has different amount of random noises introduced with probability p=0.1, 0.2, 0.6 or 1.0 where 1.0 means items are completely randomly assigned into groups without using any information from the owner attribute although the number of groups is the same as the number of unique owners. The numbers in the parentheses are the MAP or AUC improvement relative to the baseline model.

Model	Noise	Item MAP@5	Item MAP@20	Item AUC	Attribute MAP@5	Attribute MAP@20	Attribute AUC
SVDFeature	N.A.	0.0035	0.0046	0.759	N.A.	N.A.	N.A.
SVDFeature+HSM	0.0	0.0046 (+31.4%)	0.0066 (+43.4%)	0.813 (+7.1%)	0.025	0.029	0.842
	0.1	0.0045 (+28.5%)	0.0062 (+34.7%)	0.804 (+5.9%)	0.020	0.023	0.842
	0.2	0.0047 (+34.2%)	0.0064 (+39.1%)	0.794 (+4.6%)	0.016	0.019	0.830
	0.6	0.0038 (+8.5%)	0.0049 (+6.5%)	0.765 (+0.7%)	0.004	0.006	0.774
	1.0	0.0029 (-17.1%)	0.0039 (-15.2%)	0.754 (-0.6%)	0.003	0.004	0.497

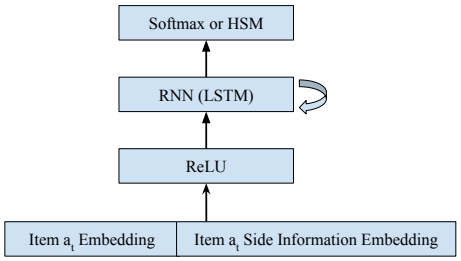

Figure 1: The architecture of the RNN model.

TF-IDF [16]. Overcoming this limitation (elaborated more later in the end) is an interesting direction of future research which could be complex by itself. We focus on initially demonstrating the efficacy of predicting item side information in this work.

This proposed approach can be applied in both the SVDFeature and RNN models. In either case, our approach does not change the user state model s. However, for SVDFeature, the item prediction model as in Equation 4 is changed to only use W_a, *i.e.*, $o(v) = W_a$, leaving out V_c for the introduced auxiliary task.

4.1 Multi-Task Learning

In the MTL approach, we predict the categorical attribute by simply sharing the same user state representation s of the item prediction model. Denote the *output-side* embedding of a categorical attribute as V. For a positive observation of a, denote its attribute ID as c. Similar to the negative sampling for a positive item, β negative attribute IDs are randomly sampled to build the following softmax prediction model in Equation 8 where c is an one-hot encoded representation of the positive and sampled negative attribute IDs:

$$g(u, o) = s^T V_c \tag{7}$$

$$\widehat{p_c}(c_i = 1) = \frac{exp(g(u, o_i))}{\sum_{k=1}^{\beta+1} exp(g(u, o_k))} \tag{8}$$

4.2 Hierarchical Softmax

In the HSM approach [18], we sequentially make predictions along a hierarchical structure of the attribute and item. The estimated probability for an item becomes the product of the estimated attribute probability associated with the item and the estimated item probability within the group of items defined by the attribute. To be more precise, let K_c be the items belonging to the attribute c's group. Then,

$$\hat{p_a} = \hat{p}_c \cdot \hat{p}_{a|c} \tag{9}$$

$$\hat{p}_{a|c}(a_i = 1|K_c) = \frac{exp(f(s, o_i))}{\sum_{k \in K_c} exp(f(s, o_k))} \tag{10}$$

In real-word applications, the grouping induced by a categorical attribute can be very unbalanced, so that for some attribute IDs, a large group of items are involved while for others only a few. To address this problem, we introduce a sub-vocabulary sampling step, *i.e.*, when the number of items involved by the attribute ID is greater than a threshold, we down-sample it to the threshold randomly. Algorithm 1 summarizes the steps training a HSM model. Algorithm 2 summarizes the HSM inference procedure after training the model.

Algorithm 1: The algorithm for training a two-level HSM model with a large unbalanced hierarchy induced by a categorical attribute.

Data: u_t, a_t, c_{a_t} for $t = 1, ..., T$, where u_t is a user history representation; a_t is the item that was acted on by the user at time step t; c_{a_t} is the associated value of a selected categorical attribute of the item.
Model Parameters: θ, W, V
Hyper Parameters: the number of attribute samples $\alpha < N$ where N is the attribute vocabulary size, the number of item samples $\beta < M$ where M is the item vocabulary size.

1 **for** *t=1,...,T* **do**
2 Run the user state model to get current user state representation $s_t(u_t, \theta)$
3 Excluding c_{a_t}, randomly sample α negative attribute samples from the attribute vocabulary.
4 Compute the negative log likelihood loss $L(c_{a_t})$ of $\widehat{p_{c_{a_t}}}$ in Equation 8.
5 Get the group of items with attribute value c_{a_t}, *i.e.* K_c.
6 Excluding a_t, randomly sample β negative item samples from the item vocabulary.
7 Compute the negative log likelihood loss $L'(a_t)$ of $\widehat{p_{a_t|c_{a_t}}}$ in Equation 1.
8 Minimize the loss $L(c_{a_t}) + L'(a_t)$ using SGD [2] or AdaGrad [9].
9 **end**
10 Repeat the *for* loop until convergence, *e.g.*, the decrease of the loss is less than certain threshold.

Result: W, V, θ

Algorithm 2: The inference algorithm of the two-level HSM model.

Data: A user history representation u; an attribute-to-items mapping K_c where $c = 0, ..., M - 1$.
Model Parameters: θ, W, V

1 Run the user state model to get current user state representation $s(u, \theta)$
2 Compute the predicted attribute probabilities $\widehat{p_c}$ as in Equation 8 for all c.
3 Compute the within-group item probabilities $\widehat{p_{a|c}}$ as in Equation 10 for all c and a.
4 Compute the final predicted item probabilities p_a as in Equation 9 for all a.

Result: p_a for all a.

5 EXPERIMENTS AND RESULTS ON BEHANCE DATA SET

In this section, we evaluate the proposed approaches of MTL and HSM for both SVDFeature and RNN models with a real-world data set: Behance [11]. Behance.net is an online community where users can create arts and design projects and share their creation with the community. It supports a feature *appreciation* through which users can express their interest on projects created by others while browsing in the site. In this application, the recommendation of projects to users can be formulated as predicting the probabilities of a user appreciating the projects in the system.

Table 3: The summary statistics of the Behance data set and the distribution of the sizes of the attribute in terms of the number of items.

Statistic	Size	Summary	Size
#users	63,497	**min**	1
#items	178,788	**25%**	1
#owners	51487	**median**	2
#appreciations	1M	**75%**	4
		max	153

The Behance data set we used was part of the data set collected and released by He *et al.* [11]. The available item side information is the *owner* or creator of the project. One project can possibly have multiple owners, in which case we take the one with the highest TF-IDF value [16] where IDF is computed across all the projects in the data set. Table 3 shows the statistics of the data set along with the distribution of the number of items belonging to the owners (a highly skewed distribution).

We employ a temporal *leave-one-out* training and testing procedure, *i.e.*, for each user, the whole sequence of appreciated items except the last one is used as the training data leaving the last one for testing. We use the metrics of Mean Average Precision (MAP@k, where k=5 and 20) and Area Under the ROC Curve (AUC). AUC is evaluating whether the models can rank the last appreciated item of a user (*i.e.* positive item) higher than a negative item randomly sampled from all the items in the data set (excluding the last appreciated item of the user).

We implemented all models based on TensorFlow [1] (Version 1.4.1, Python API; open sourced in GitHub [1]). For all the models, we use embedding dimension $d = 32$. They are trained until convergence by running the AdaGrad algorithm with initial learning rate $\eta = 1.0$ (we found this value better than the default $\eta = 0.1$). The batch size of the AdaGrad minimization is 32, *i.e.*, each batch has 32 users' appreciation sequences which involves on average 400 positive items. We use $\alpha = \beta = 2K$ for each batch, *i.e.*, sampling 2K negative items (excluding the positive items) for each batch in the item prediction softmax, attribute prediction softmax and the sub-vocabulary sampling of the HSM model.

Table 1 shows the results. We see that both MAP@k and AUC show consistent substantial improvement for HSM on top of both SVDFeature and RNN. MTL shows substantial improvement in terms of MAP@k but more improvement is observed on top of SVDFeature than RNN. Interestingly, we observe that modeling the user history sequence based on RNN is substantially better than not modeling the sequence based on SVDFeature but the improvement only manifests in terms of MAP@k, not in terms of AUC. Although it is also possible to model the user history sequence through embedding vector summation in SVDFeature, we chose not to here because we want to see whether MTL or HSM can improve on top of substantially different user models.

The MAP@k and AUC are much higher for the task of attribute prediction than item prediction, which suggests that predicting

[1] https://github.com/grouplens/samantha

higher-level groups induced by the attributes are easier tasks than directly predicting the lower-level item confirming with our hypothesis at the beginning.

Table 2 shows how accuracy improvement is affected when noise is introduced into the group assignment of items based on attributes. When the noise $p = 1.0$, *i.e.*, all items are randomly assigned into a group, the HSM model achieves similar but slightly worse performance than the baseline model, which shows that adding additional random grouping in the output hurts probably because the attribute model struggles to learn a coherent embedding for that group of items while those items are not similar to each other at all.

Table 4: The summary statistics of the large scale industrial data set and the distribution of the sizes of the attribute in terms of the number of items.

Statistic	Size	Summary	Topic Size	Publisher Size
#users	Hundreds of Millions	**min**	1	1
#items	2M	**25%**	1	1
#topics	600K	**median**	1	1
#publishers	800K	**75%**	3	2
#consumptions	Hundreds of Billions	**max**	38K	3.5K

6 IMPROVING RNN-BASED LARGE-SCALE INDUSTRIAL PRODUCTION MODEL

We further evaluate the proposed approach on top of a large-scale private industrial production model (RNN-based). Table 4 lists the statistics of the data set and the distribution of the sizes of each attribute used. The categorical attributes involved here are *publisher* and *topic*. Each item in the data set is annotated with the most relevant topic ID and each item has an unique publisher ID. They are all embedded as the input of the RNN as shown in Equation 5 along with other input features used in the production model. As Table 4 shows, the distribution of group sizes induced by these attributes are extremely unbalanced.

The embedding dimension $d = 256$ and the initial learning rate $\eta = 0.1$. Because of the size of the item space, we use 20K negative samples for the item prediction softmax and attribute prediction softmax. For the conditional item prediction task in the HSM model, we limit using 400 negative samples for each attribute ID in the batch (batch size is 128), in which case the effective number of items involved each batch is 6K for the topic attribute and 2K for the publisher attribute.

We employed a similar temporal *leave-one-out* training and testing procedure in this data set. Table 5 shows the item prediction accuracy measured by MAP@k (k=5 and 20). We see that both MTL and HSM improve the recommendation accuracy, especially HSM, which has around 20% accuracy gain across the two types of attributes for MAP@5 while MAP@20 shows consistent but slightly less accuracy improvement. MTL seems to depend on which attribute to use because the improvement using the topic is much larger (7.9% gain) than using the publisher (3.3% gain).

Table 6 shows how MAP changes when we introduce noises into the item group assignment based on the attribute publisher running the HSM model. It shows consistent results with the experiments run on the Behance data set.

7 ANALYSIS AND SIMULATION

In this section, we investigate how introducing the auxiliary task of predicting categorical attributes can help item prediction. We focus on HSM here because it has more consistent and substantial improvement. We did not come up with a theoretical explanation but conducted analysis on the Behance data set and simulation experiments to test a few hypotheses.

H1: Does hierarhical softmax have more improvement for the long-tail items' prediction (Yes). To answer this question, we analyze the relationship between the popularity of the two items (one positive and one randomly sampled negative) to be predicted for a user (along with the popularity of their attributes) and whether SVDFeature+HSM is better than SVDFeature (*i.e.*, SVDFeature+HSM ranks correctly while SVDFeature ranks incorrectly, referred to as the *Better* variable here) based on the Behance data set. Popularity here refers to the number of times an item or attribute appears in the appreciation data set. We use the rank of an item or attribute's popularity instead of the raw number. Table 7 shows the coefficients of the popularity of items and attributes predicting the Better response variable in a logistic regression model. We see that the effects of item popularity are significant and their signs are positive (note that higher values of popularity rank represent less popularity, *i.e.* the tail). That the effect of attribute popularity is not significant suggests the advantage of HSM does not favor more popularity attributes.

H2: does the advantage of HSM generalize across different types of generative models of users? (Yes) One hypothesis we have regarding HSM is that it might better fit how users express interest and discover items, *i.e.*, users have hierarchical interest (and shifting) along the hierarchy of attributes and items and HSM better models that. To investigate this question, we simulated two data sets of 1K users based on two types of generative models: *Single-Level* versus *Two-Level* within the framework of matrix factorization. Algorithms 3 and 4 show how user interactions are generated. Note that k_u and k_a control the skewness of the user interest distribution in the attributes (a user's embedding vector is modeled as a sparse distribution on the attribute embedding vectors) and the sizes of the attributes in terms of the number of items.

After generating the simulated data sets, we trained two models SVDFeature and SVDFeature+HSM. The training process is exact the same as we trained the models for the real-application data sets. Table 8 shows the accuracy results in terms of AUC of the two models ($d = 32$, negative sampling $\alpha = \beta = 200$). We observe that HSM not only improves substantially for the two-level data set, whose generative process is a better fit of the HSM model, but also for the single-level data set. This analysis suggests that overall HSM based on informative item grouping is a more powerful modeling technique than one-level flat softmax generalizing across environments of different user models of preferences and behaviors.

8 DISCUSSION, LIMITATION AND FUTURE WORK

Our experimental results demonstrated that using items' categorical attributes through either MTL or HSM can benefit item prediction in recommendation problems. Particularly for HSM, from the softmax normalization perspective, our results suggest that the negative

Table 5: Results for the large scale data set: The testing MAP@k (k=5 or 20) of the item and attribute prediction tasks for MTL and HSM on top of RNN. The numbers in the parentheses are the MAP improvement relative to the baseline RNN model. Best item MAP numbers are bolded.

Model	Attribute	Item MAP@5	Item MAP@20	Attribute MAP@5	Attribute MAP@20
RNN	N.A.	0.151	0.171	N.A.	N.A.
RNN+MTL	Topic	0.163 (+7.9%)	0.178 (+4.0%)	0.336	0.349
	Publisher	0.156 (+3.3%)	0.174 (+1.7%)	0.293	0.309
RNN+HSM	Topic	**0.184 (+21.8%)**	**0.197 (+15.2%)**	0.337	0.351
	Publisher	0.182 (+20.5%)	**0.197 (+15.2%)**	0.295	0.310

Table 6: Results for the large scale data set: The testing MAP@k (k=5 or 20) of the item and publisher prediction tasks for the baseline model and HSM model whose group assignment has different amount of random noises introduced with probability p=0.1, 0.2, 0.6 or 1.0 where 1.0 means items are completely randomly assigned into groups without using any information from the publisher attribute although the number of groups is the same as the number of unique publishers. The numbers in the parentheses are the MAP improvement relative to the baseline RNN model.

Model	Randomization	Item MAP@5	Item MAP@20	Publisher MAP@5	Publisher MAP@20
RNN	N.A.	0.151	0.171	N.A.	N.A.
RNN+HSM	0.0	0.182 (+20.5%)	0.197 (+15.2%)	0.295	0.310
	0.1	0.168 (+11.2%)	0.182 (+6.4%)	0.269	0.281
	0.2	0.166 (+9.9%)	0.180 (+5.2%)	0.250	0.265
	0.6	0.152 (+0.6%)	0.165 (-3.5%)	0.183	0.196
	1.0	0.150 (-0.6%)	0.163 (-4.6%)	0.151	0.163

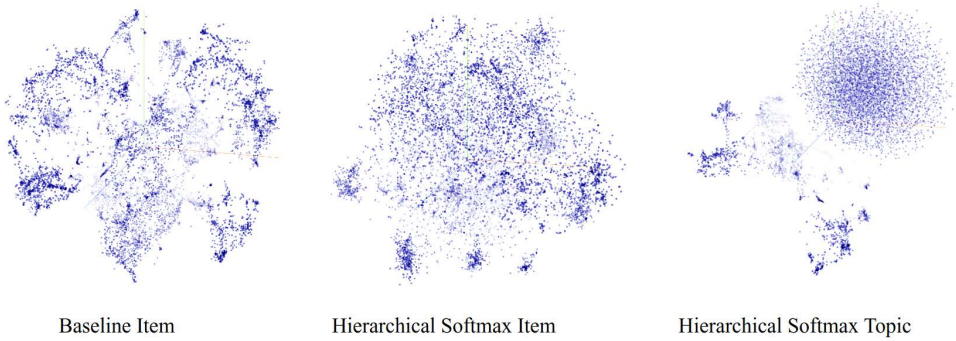

Figure 2: The t-SNE embeding visualization of the item embeddings and topic embeddings for the single-task baseline and hierarchical softmax models.

Table 7: The coefficients (standard errors) of the popularity of items and attributes predicting whether SVDFeature+HSM is better than SVDFeature in a logistic regression model. Each user has one observation. ***p<0.001.

Popularity	Coef. (Std.)
log(true item rank)	0.613 (0.012) ***
log(false item rank)	0.051 (0.008) ***
log(true attribute rank)	0.004 (0.011)
log(false attribute rank)	0.008 (0.009

Table 8: The AUCs of SVDFeature and SVDFeature+HSM on top of two simulated data sets based on two types of generative user models.

Data Set	Model	AUC
Single-Level	SVDFeature	0.592
	SVDFeature+HSM	0.668 (+12.8%)
Two-Level	SVDFeature	0.652
	SVDFeature+HSM	0.705 (+8.1%)

samples used for normalization in Equation 10 are more effective than the random negative samples in Equation 1. Given that the higher-level differences of the items are handled by the higher-level attribute prediction task, the item-related model parameters can focus on learning more nuanced patterns within the attribute group. Figure 2 is an illustration of the t-SNE [15] embeddings of the item embedding in the baseline RNN model and the item, topic embeddings in the RNN+HSM model. We see that the topic embeddings encode some large-scale grouping structure in the item

Algorithm 3: The simulation algorithm for generating item-attribute mapping and item/attribute embeddings for generating user history interactions in Algorithm 4. $\sim_i$ represents sampling i times according to certain distribution.

Para: item vocabulary size $N = 5000$; attribute vocabulary size $M = 500$; the shape of the Gamma distribution for item assignment in attributes $k_a = 1.0$. Embedding dimension $d = 32$

#generating assignment of items to attributes.
$\alpha \in R_M \sim_M Gamma(shape = k_a, scale = 1.0)$
$C \in Z_M$
for *i=1,...,N* **do**
  $\phi \in R_M \sim Dirichlet(\alpha)$
  $C_i \sim Categorical(\phi)$
end
#generating attribute embedding weights.
$V \in R_{Mxd} \sim Uniform(0, 1)$
#generating item embedding weights.
$W \in R_{Nxd} \sim_N Gaussian(V_{C_a}, 0.01), a = 1 : N$
Result: W, V, C

Algorithm 4: The simulation algorithm for sampling the interaction history of users. $\sim_i$ represents sampling i times according to certain distribution.

Para: W, V, C from Algorithm 3; the shape of the Gamma distribution for sampling user interest in attributes $k_u = 1.0$; embedding dimension $d = 32$; the number of users to generate $m = 1000$; the number of items (and their corresponding attributes) to generate n for each user; the type of the generative model *type*: Single-Level or Two-Level

for *i=1,...,m* **do**
  #generating user embedding weights.
  $\alpha \in R_M \sim_M Gamma(shape = k_u, scale = 1.0)$
  $\phi \in R_M \sim Dirichlet(\alpha)$
  $u = V^T \phi$
  #generating the item according to user attribute interest.
  if *type is Single-Level* **then**
    $a \in Z_n \sim_n Softmax(Wu)$
    $c \in Z_n$, where $c_j = C_{a_j}, j = 1 : n$
  end
  else if *type is Two-Level* **then**
    $c \in Z_n \sim_n Softmax(Vu)$
    $a \in Z_n$, where $a_j \sim Softmax(W_{c_j} u)$, W_{c_j} is the embeddings of the items involved by $c_j, j = 1 : n$.
  end
end
Result: the m tuples of (i, a, c)

vocabulary. The item embeddings in the baseline RNN model seem to be more clustered than the RNN+HSM model.

In this work, the classification models are limited to utilize only one categorical attribute, we consider it as interesting future work to combine multiple attributes through multi-layer hierarchical softmax or multi-task learning predicting multiple dimensions of the item space to further improve. A challenge is that not only the item-attribute dependency needs to be modeled, but also the inter-dependencies among multiple attributes, *e.g.*, given that a user likes a topic, what is the probability that the user likes the publishers within that topic, or alternatively, that the user likes the topics within particular publishers (note that a publisher is the user who created an item in the system). It is not intuitive to decide what is the right dependency order. In either case, this hierarchical prediction is inherently sequential, which resembles the gradient boosting techniques to some extent [10].

In terms of applications, the proposed classification model is a natural fit for grouped or categorized recommendation presentation interfaces. For example, many recommender systems present their recommendations grouped by topics, genres or categories instead of using a combined top-N list. Besides, outside of recommender systems, in any interface that supports browsing by topics or other criteria can use the HSM model to personalize the browsing experience given that a user clicks certain topic link. Lastly, since these attributes have nice interpretability from the user's perspective, the model may be used for eliciting user preferences interactively in a conversational recommender system or the coarser-level action space exploration in reinforcement learning for recommender systems.

9 CONCLUSION

In this work, we propose models utilizing the readily available item categorical attributes as output for multi-class classification-based recommender systems. In our proposed models, we introduce structures induced by the item categorical attributes in the classification softmax either through multi-task learning or hierarchical softmax.

In the multi-task learning case, along with the task of predicting the next item that a user will act on, we also predict the higher-level attribute value of the next item but still sharing the same user state model, hypothesizing that this auxiliary task can benefit the item model learning through transfer learning or regularization effects.

In the hierarchical softmax case, we not only introduce the same auxiliary task sharing the same user state model, but also condition our item-level prediction task on the attribute-level prediction output.

We demonstrate through experiments on real-world data sets that these two approaches improve item-level prediction substantially on top of the modeling frameworks of both matrix factorization and neural networks. We demonstrate the robustness of the hierarchical softmax approach by introducing noise in building the hierarchy and the its generalizability across different user generative models through simulation experiments.

In summary, we show that introducing predicting categorical attributes as auxiliary tasks in item prediction has substantial advantages for recommender applications. Despite unbalanced hierarchies, in practice, the approach is relatively easy to apply and fairly robust against noise in the categorical grouping. We hope this result is particularly useful to other recommender practitioners.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[3] Rich Caruana. 1998. Multitask learning. In *Learning to learn*. Springer, 95–133.

[4] Rich Caruana and Virginia R De Sa. 1997. Promoting poor features to supervisors: Some inputs work better as outputs. In *Advances in Neural Information Processing Systems*. 389–395.

[5] Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao, and Yong Yu. 2012. Collaborative personalized tweet recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 661–670.

[6] Tianqi Chen, Linpeng Tang, Qin Liu, Diyi Yang, Saining Xie, Xuezhi Cao, Chunyang Wu, Enpeng Yao, Zhengyang Liu, Zhansheng Jiang, et al. 2012. Combining factorization model and additive forest for collaborative followee recommendation. *KDD CUP* (2012).

[7] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research* 13, Dec (2012), 3619–3622.

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[9] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[10] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[11] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: a visually, socially, and temporally-aware model for artistic recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 309–316.

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[15] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[16] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Scoring, term weighting and the vector space model. *Introduction to information retrieval* 100 (2008), 2–4.

[17] Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*. 1081–1088.

[18] Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model.. In *Aistats*, Vol. 5. 246–252.

[19] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[20] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[21] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[22] Shilad Sen, Jesse Vig, and John Riedl. 2009. Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web*. ACM, 671–680.

[23] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.

[24] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine Learning* 73, 2 (2008), 185–214.

[25] 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*. ACM, 495–503.

[26] Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng. 2011. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 295–304.