Taxonomy Discovery for Personalized Recommendation

Yuchen Zhang
UC Berkeley
Berkeley, CA, USA
yuczhang@berkeley.edu

Amr Ahmed, Vanja Josifovski
Google Inc
Mountain View, CA, USA
{amra,vanjaj}@google.com

Alexander Smola
Carnegie Mellon University
Pittsburgh, PA, USA
alex@smola.org

ABSTRACT

Personalized recommender systems based on latent factor models are widely used to increase sales in e-commerce. Such systems use the past behavior of users to recommend new items that are likely to be of interest to them. However, latent factor model suffer from sparse user-item interaction in online shopping data: for a large portion of items that do not have sufficient purchase records, their latent factors cannot be estimated accurately.

In this paper, we propose a novel approach that automatically discovers the taxonomies from online shopping data and jointly learns a taxonomy-based recommendation system. Out model is non-parametric and can learn the taxonomy structure automatically from the data. Since the taxonomy allows purchase data to be shared between items, it effectively improves the accuracy of recommending tail items by sharing strength with the more frequent items. Experiments on a large-scale online shopping dataset confirm that our proposed model improves significantly over state-of-the-art latent factor models. Moreover, our model generates high-quality and human readable taxonomies. Finally, using the algorithm-generated taxonomy, our model even outperforms latent factor models based on the human-induced taxonomy, thus alleviating the need for costly manual taxonomy generation.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles; G.3 [**Mathematics of Computing**]: Probability and Statistics

General Terms

Algorithms, Theory, Experimentation

Keywords

Recommender System, Latent Factor Model, Taxonomy

WSDM'14, February 24–28, 2014, New York, New York, USA.
ACM 978-1-4503-2351-2/14/02.
http://dx.doi.org/10.1145/2556195.2556236.

1. INTRODUCTION

Personalized recommender systems are used widely to increase sales and customer satisfaction in e-commerce. These systems use past behavior of users to recommend new items that are likely to be of interest to them. One of the most extensively studied techniques are latent factor models and related variants [1, 8, 9, 10]. These models project users and items into a lower-dimensional space of latent factors. Then, the similarity between a particular user and an item is computed via the inner product of their latent factors and the most similar items are recommended to the user.

Despite substantial success in the Netflix contest [3, 8], in tag recommendation [15] and other applications, the latent factor model approach encounters specific challenges in product recommendation for online shopping: on a typical retail website we observe a long tail effect both in terms of users and in terms of items. This means that not only most users only buy a small number of items, but also that the majority of items are only infrequently purchased. On our data users typically purchase 2.4 items and 85% of the items are purchased by less than 10 users. This sparsity of user-item interactions makes it difficult to learn latent factors.

In order to resolve the sparsity problem in online shopping, Kanagal et al. [7] use a human-induced taxonomy that attaches every item to a node in the category tree. Their proposed taxonomy-aware latent factor model assumes that the latent factor associated with each tree node is sampled from its parent node, thus generalizing the purchase data from an individual item to items belonging to the same category. Experimental studies show that this significantly improves the performance for online shopping data. Other related work includes Mnih et al. [12] and Menon et al. [11] who use the taxonomy to measure biases in music recommendation, and earlier works by Ziegler et al. [17] and Weng et al. [16] who incorporates the taxonomy in alternative recommender systems besides latent factor models.

Unlike previous work that requires an existing taxonomy, we propose in this paper a novel approach that automatically discovers the taxonomy from online shopping data and jointly learns a taxonomy-based recommendation system from raw data. This has several benefits:

1. Many online shopping datasets don't have an associated human-induced taxonomy since it may be too expensive to create a hand-crafted taxonomy and attach every item to the category. In contrast, for our algorithm it is sufficient to obtain the textual description text of items (whenever available) and purchase records. Both sources are much easier to collect.

2. Human-created taxonomies are static and they do not evolve with a change in user demographics or product inventory. Our method adaptively handles incremental data. This makes it capable of dynamically updating the tree structure.
3. Human labelings are noisy and not optimized for learning latent factor models. For example, in our reference taxonomy, video game software and video game consoles are categorized into two different top-level departments: "Software" and "Electronics". In reality, they are usually bought together and thus have similar latent factors. Our method combines categorization with latent factor optimization so that items having similar latent factors tend to have common ancestors in the tree, which allow them to share purchase data.

We use a nonparametric generative procedure which we refer to as a hierarchical latent factor model (HF Model). It generates a tree structure for categories and items by means of a nested Chinese Restaurant Process (nCRP) [4]. Then, the item's descriptions are generated via a language model and the user purchase data is generated by a latent factor model. We propose an inference algorithm to jointly optimize the tree structure and the latent factors. Thereby the HF model constructs a taxonomy that depends on both the item's description and on the user's purchase data.

Recent work of Mnih et al. [13] clusters items a binary tree by their latent factors in order to reduce the computation complexity of inference. Agarwal et al. [1] propose a regression-based latent factor model that uses meta-information to help tail items. We found that building a dynamic taxonomy based on both descriptions and purchase data performs considerably better than these models.

We report comprehensive experiments that compare the HF model with state-of-the-art latent factor models on a large-scale online shopping dataset. There the HF model significantly improves on existing models. We also compare the HF model to the latent factor model that relies on human-induced taxonomy. We find that although the HF model requires no human effort, it outperforms the human dependent model. An extensive study shows that the HF model generates high-quality and human-readable taxonomies.

The rest of this paper is organized as follow. In Section 2, we describe related works and baseline models. In Section 3, we define the hierarchical latent factor model with its learning algorithm described in Section 4. In Section 5, we present experiments that compare the HF model with state-of-the-art latent factor models. In Section 6, we illustrate how the HF model interacts with human-induced taxonomy, and conduct empirical studies to illustrate its advantage over methods that directly use the human-induced taxonomy as a priori.

2. RELATED WORKS

We give a brief overview of four state-of-the-art latent factor models for personalized recommendation, since we will compare their performance to the HF model empirically.

2.1 Latent Factor Models

A latent matrix factorization model (MF) assumes that there is a factor associated with each user and each item. We denote the factor for user u and item i by $v_u, v_i \in R^d$ respectively. Each item is associated with a bias term b_i that models the relative popularity of the item (we need no such bias for users since we recommend items to users rather than the converse). The model defines an affinity score x_{ui} between user u and item i by an inner product, that is:

$$x_{ui} = \langle v_u, v_i \rangle + b_i.$$

When the observed user-item interaction is in the form of an implicit feedback where we only have access to positive interactions between users and items, we adopt the Bayesian Personalized Ranking (BPR) criterion as described in [14]. BPR adopts a preference ranking approach by ranking items that the user liked (or bought) higher than items that the user did not interact with. For items i and j we denote by R_{uij} the event of user u preferring i to j. In this case

$$P\left(R_{uij} | v_u, v_i, v_j\right) = \sigma\left(x_{ui} - x_{uj}\right) \text{ with } \sigma(x) = \frac{1}{1+e^{-x}}.$$

We denote the set of model parameters (item and user factors, biases) by Θ and we summarize the pairwise relations by R. Based on the above model the likelihood of the dataset factorizes via

$$P(R|\Theta) = \prod_{u \in U} \prod_{i \in B_u} \prod_{j \notin B_u} P(R_{uij} | v_u, v_i, v_j).$$

Here U is the set of all users and B_u is the set of items that user u purchases. If we further assume that every entry in Θ is independently sampled from a normal distribution $\mathcal{N}(0, 1/\lambda)$, then the log-posterior distribution of parameters given the data is represented by

$$\log P(\Theta|R) = \sum_{u \in U} \sum_{i \in B_u} \sum_{j \notin B_u} \log \sigma(x_{ui} - x_{uj}) - \lambda \|\Theta\|_2^2. \quad (1)$$

The Bayesian Personalized Ranking (BPR) method estimates Θ by minimizing the negative log posterior (1) via stochastic gradient descent (SGD). For each iteration, a user u, a purchased item i and an unpurchased item j are sampled. The gradient of the associated log-posterior contribution is calculated and the parameters are updated via SGD.

2.2 Collaborative Item Selection Model

The collaborative item selection model (CIS) [13] organizes items in a binary tree, where the leaf nodes represent items and the internal nodes represent intermediate categories. Let n be a node of the tree and $C(n)$ be the set of its children. The CIS model assumes that for user u, the probability of moving from node n_p to node n_c on a root-to-leaf tree traversal is given by

$$P(n_c | n_p, u) = \frac{\exp(U_u^\top Q_{n_c} + b_{n_c})}{\sum_{m \in C(n_p)} \exp(U_u^\top Q_m + b_m)} \text{ if } n_c \in C(n_p).$$

Here Q_n and b_n are factors and biases of node n, and U_u is the factor vector of user u. The probability of selecting item i is then given by the product of the probabilities of the decisions leading from the root to the leaf containing i:

$$P(i|u) = \prod_{j=1}^{L_i} P(n_j^i | n_{j-1}^i, u).$$

Here n_j^i represents the j-th node in the path from the root to item i. Given a tree over items, the CIS model can be trained using stochastic gradient ascent in log-likelihood, updating parameters after each user/item pair. Mnih et al. [13] also propose an approximate algorithm to learn the structure of the tree. Their approach assumes that the latent factors are known. It then learns individual levels of the tree iteratively from top to bottom by maximizing log-likelihood. The model is trained in a three-stage procedure. First, we train a CIS model based on a random balanced binary tree, then extract the user vectors learned by the model and use them to learn a better tree from the data. Finally, we train a CIS model based on the learned tree, updating all the parameters, including the user vectors.

2.3 Regression-based Latent Factor Model

The regression-based latent factor model (RLFM) [1] maps features of items and users into a lower dimensional factor space and then defines the affinity between users and items using these latent factors. Concretely, let v_{uj}, w_u and z_i be dyadic, user and item feature vectors, then the affinity score x_{ui} is given by

$$x_{ui} = b^\top v_{ui} + w_u^\top g + d^\top z_i + w_u^\top G^\top D z_i$$

where the parameters of the model are the vectors b, g and d, and matrices G and D. Given affinity scores, [1] defined a set of response functions to model the observed data and learn the model parameters by maximizing the log-likelihood of the observed data. However since in our problem we are only given implicit feedback, i.e. only positive interaction, we adopt the BPR objective function as in Section 2.1 in training the RLFM. For simplicity of notation we will refer to this BPR-modified variant as RLFM throughout the paper, since all models except CIS are trained using the BPR objective. We learn the RLFM model parameters by maximizing log-likelihood through stochastic gradient ascent. Note that within feature vectors w_u and z_i, we always include the unique IDs of the user and the item. If there are other meta-information associated with either user or item, we concatenate them to the feature vector.

2.4 Taxonomy-aware Latent Factor Model

To address prediction accuracy issues related to cold-start for new items and the problem of sparsity, taxonomy-based models were proposed. The main idea is to utilize the categorical information in a human-induced taxonomy to share statistical strength between frequently purchased items and tail items.

In the taxonomy-aware latent factor model (TF) [7], a latent variable is associated with each user u, each item i, and each category k. Specifically, we use w_u, w_i and w_k to represent the corresponding factors. Given an item or a category i, let $\pi(i)$ indicate the parent of i in the taxonomy tree. Then, the latent factor for item i is defined recursively

$$v_i = \begin{cases} w_i & i \text{ is the root.} \\ w_i + v_{\pi(i)} & \text{otherwise.} \end{cases}$$

In other words, the effective factor associated with node i is the sum of all latent factors associated with nodes along the path from (and including) i to the root. The affinity score between user u and node i is consequently defined by:

$$x_{ui} = \langle w_u, v_i \rangle + b_i.$$

As above, the BPR objective is used for model estimation.

Table 1: Notation used for the HF model.

u	User
i, j	Item
v_u, v_i	Latent factor for user u or items i
x_{ui}	Affinity score between user i and item i
y, z	Internal nodes of the categorization tree
$\pi(z)$	Parent of node z
$C(y)$	Children of node z
ϕ_z, φ_z	Multinomial distribution associated with z
n_z	Number of items belonging to category z
α	Chinese Restaurant Process parameter
β, η	Dirichlet distribution parameter
D_i, A_i	Description of item i (a set of terms)
t, a	Term in the description
q_i	Popularity measure of item i
σ^2, τ^2	Variance for generating latent variables.

3. HIERARCHICAL LATENT FACTOR MODEL

Our goal in designing a hierarchical latent factor model (HF) is to automatically generate a hierarchical categorization for all items based on their descriptions and their purchase data, so that we can learn the item/user latent factors jointly. Our model has the following features:

- It arranges the items in a taxonomy with infinite (adaptive) depth and width using a non-parametric prior [2].
- In addition to purchase data it can use side-information such as item descriptions to infer the taxonomy.
- It smoothes the model parameters over the induced taxonomy and as such combats data sparsity.
- It can utilize a partial (or full) human-induced taxonomy when available (we defer details to section 6).

Table (1) summarizes the notation used in the paper.

3.1 Taxonomy Generation

We now describe a non-parametric prior over trees with infinite depth and width. This is similar to the nested Chinese restaurant process in [2, 4]. In a nutshell, to organize items in a tree, one needs to generate a path for each item over the tree. A path can be conceptually viewed as a *nested* set of decisions. Starting from the root, a child is selected and the process continues until a termination condition is satisfied. These choices can be modeled using a Chinese Restaurant Process (CRP) where at each node the probability of selecting a child is proportional to the child's frequency.

Consider an arbitrary item i that we want to attach to the category tree. For this to happen we generate a path in the tree from the root to the leaf node that represents i. Starting from node v, the probability of selecting an existing (or new) child is

$$P(y \to z) = \begin{cases} \frac{n_z}{n_y+\alpha} & \text{if } z \in C(y) \\ \frac{\alpha}{n_y+\alpha} & \text{for a new node} \end{cases} \tag{2}$$

Recall that n_y is the number of items belongs to category y and n_z is the number of items that belongs to z. The parameter α controls the probability of creating a new child for y, i.e. the probability that the item belongs to a new category under y.

Once a child node is selected, the process is repeated until a full path is defined. To ensure finite paths we need to allow for the probability of termination at a vertex. Here, we define that the process terminates at category z if z is the first child of its parent. This strategy is in complete analogy to Ghahramani et al. [6] — we treat the probability of terminating at a vertex in complete analogy to that of generating a special child.

3.2 Parameter Cascade

The nCRP process gives a distribution over trees. However, we still need to assign parameters to each node in the tree and tie these parameters in a manner that is consistent with the semantic of the tree, i.e. nodes close in the tree should have similar parameters. We endow each internal node z with two latent variables: a latent factor v_z and a multinomial distribution over terms ϕ_z. These parameters are cascaded over the tree as follows:

$$v_z \sim \begin{cases} \mathcal{N}(0, \sigma^2 \mathbf{1}) & w \text{ is the root node} \\ \mathcal{N}(v_{\pi(z)}, \sigma^2 \mathbf{1}) & \text{otherwise} \end{cases} \tag{3}$$

and the multinomial is sampled by a Dirichlet distribution:

$$\phi_z \sim \begin{cases} \mathrm{Dir}(\beta) & z \text{ is the root node} \\ \mathrm{Dir}(\eta \phi_{\pi(z)}) & \text{otherwise} \end{cases} \tag{4}$$

3.3 Generating item data

After we generate the tree with its associated parameters, we need to generate data associated with item i: an item description, an item latent factor and item bias. Item latent factors are sampled from its parent's latent factors via

$$v_i \sim \mathcal{N}(v_{\pi(i)}, \sigma^2 \mathbf{1}).$$

We generate the item description according to the multinomial distribution associated with its parent. In particular, for each term t in the description of item i, it is sampled by a multinomial distribution:

$$t \sim \mathrm{Mult}(\phi_{\pi(i)}). \tag{5}$$

Every item also maintains a popularity measure q_i. A high q_i indicates that the item attracts customer regardless of the customer's latent factor. The popularity measure is generated by a normal distribution

$$q_i \sim \mathcal{N}(0, \tau^2). \tag{6}$$

The generative procedure for items and categories is complete by combining (2)-(6) appropriately.

3.4 Generating User Purchase Preferences

Given items and their features obtained in Section 3.3, we can then generate the purchase preference R_{uij} for any particular user u and item pairs i, j via the BPR model. For an arbitrary item i, we define the affinity score between u and i to be

$$x_{ui} = \langle v_u, v_i \rangle + [q_i]^+$$

Here, v_u and v_i are user and item latent factors, q_i is the popularity measure that we define in Section 3.3. The notation $[q_i]^+ = \max(q_i, 0)$ indicates that we force the contribution from the popularity measure to be non-negative. We impose this constraint because $[q_i]^+$ behaves like a bias term in the latent factor model. Consequently, its value is strongly correlated to the frequency that item i is purchased in the user log data. Without the $[\cdot]^+$ operator, infrequently purchased items will always have negative biases, which makes

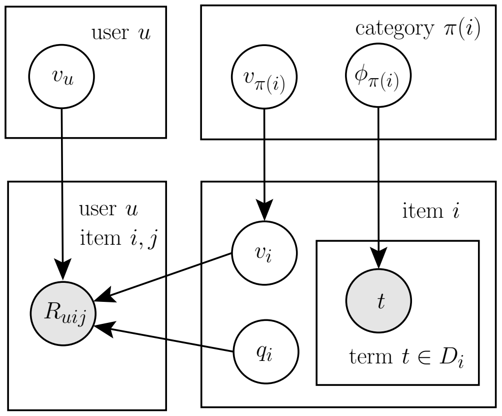


Figure 1: Graphical model representation of the H-F model. White nodes represent random variables and shaded nodes represent observations. R_{uij} is the event that user u prefers item i over item j. Large plates indicate parts of the graph that are repeated. Graphical model omits some dependencies to avoid cluttering the display. For example the hidden variable $\pi(i)$ is sampled from nCRP process. Moreover, dependencies between categories and subategories latent variables ϕ and v are omitted for clarity. See Section 3 for a full description.

them unlikely to be recommended to any customer. Since a large percentage of items are infrequent, this makes the personalized recommendation ineffective. By adopting the non-negative constraint on q_i, it promotes frequent items but does not penalize infrequent items. This allows the model to recommend tail items to specific groups of target customers as long as their latent factors match. This yields

$$R_{uij} \sim \mathrm{Bernoulli}(\sigma(x_{ui} - x_{uj})).$$

An alternative means of deriving R_{uij} is as follows: Given the affinity x_{ui}, the probability that u selects i is given by

$$P(i|u) = \frac{\exp(x_{ui})}{\sum_j \exp(x_{uj})}.$$

By this definition, for two items i and j available to u, the probability that the user chooses i over j is

$$P(i > j|u) = \frac{\exp(x_{ui})}{\exp(x_{ui}) + \exp(x_{uj})} = \sigma(x_{ui} - x_{uj}) \tag{7}$$

Note that the conditional probability (7) is consistent with the BPR optimization criterion [14]. Figure 1 shows a simplified summary of the HF model.

4. INFERENCE

We observe item descriptions and purchase data. The goal of learning is to infer the tree structure, category parameters and latent factors. We use a collapsed Gibbs sampler and we integrate out multinomial variables ϕ to improve mixing. Our goal is thus to infer the posterior $P(\Theta, T|D, R)$, where T denotes the tree structure, Θ denotes the remaining latent variables (factors and biases for items and categories), D denotes item description and R purchase preferences. We alternate until convergence between two steps: sampling a path for each item over the tree and then optimizing the latent factors of items and categories while keeping the tree structure fixed. The following sections describe each step.

4.1 Sampling Hierarchical Categories

Sampling a tree structure T amounts to sampling a path for each item i over the tree. Collectively the set of item paths defines the tree. We denote by $p_i = (p_{i0}, p_{i1}, \cdots)$ the path of item i, where p_{i0} is the root, p_{i1} is a child of the root selected by item i, etc.; In general $p_{i,k} \in C(p_{i,k-1})$. If $l(p_i)$ denotes the length of the path, then $p_{i,l(p)}$ is the parent category of item i. The probability that the path is sampled is given by:

$$P(p_i|D_i, v_i, \text{rest}) = P(p_i|\text{rest})P(v_i, D_i|p_i, \text{rest}) \quad (8)$$

where rest denotes all other hidden variables. The first component defines the prior probability of the path, while the second component defines the likelihood of item description and latent variable given this path selection. The prior probability of a path p_i is given by the nCRP process as follows $P(p_i|\text{rest}) = \prod_{k=0}^{l(p_i)-1} P(p_{i,k} \to p_{i,k+1})$, where each factor is determined by (2).

Unfortunately this is very costly, since the space of possible paths scales as $O(N)$, where N is the number of nodes in the tree. Therefore we resort to a greedy approximation following [2] which works well in practice. In this approximation we use a level-wise strategy: assume that we are at level k and that $p_{i,k} = y$, then we can descend the tree as follows:

1. Stay on the current node y — i.e. pick child 0, and set $p_{i,k+1} = 0$.
2. Move to a child node z of y other than child 0, and set $p_{i,k+1} = z$.
3. Create a new child node of node y and move to it, and set $p_{i,k+1}$ accordingly.

The probability of each choice shares a similar form:

$$P(p_{i,k+1} = z|p_{i,k} = y, \text{rest})P(D_i, v_i|p_{i,k+1} = z, \text{rest}) \quad (9)$$

Here the first probability is

$$P(p_{i,k+1} = z|p_{i,k} = y, \text{rest}) = P(y \to z) \quad (10)$$

as defined in (2).

The second term is essentially the probability of the item data given a choice of the parent z under consideration in the path which can be decomposed into two components: the probability of the item description $P(D_i \,|\, p_{i,k+1} = z, \text{rest})$, and the probability of the item factor given its parent's factor $P(v_i \,|\, p_{i,k+1} = z, \text{rest})$. The latter probability is simply normally distributed:

$$P(v_i \,|\, p_{i,k+1} = z, \text{rest}) = \mathcal{N}(v_z, \sigma) \quad (11)$$

Since we integrated out the multinomial distributions ϕ, computing $P(D_i \,|\, p_{i,k+1} = z, \text{rest})$ amounts to a standard Dirichlet-multinomial integration, that is:

$$P(D_i \,|\, p_{i,k+1} = z, \text{rest}) = \prod_{t \in D_i} \frac{\overline{m}_{z,t}^{-i}}{\overline{m}_z^{-i}}. \quad (12)$$

where the notation $\overline{m}_{z,t}^{-i}$ indicates the *regularized* occurrence of term t in the item descriptions under category z (excluding item i). In particular, we have

$$\overline{m}_{z,t}^{-i} = \begin{cases} m_{z,t}^{-i} + \beta & v \text{ is the root node.} \\ m_{z,t}^{-i} + \eta \cdot m_{y,t} & \text{otherwise} \end{cases} \quad (13)$$

where $m_{z,t}^{-i}$ is the exact occurrence of term t under category z (excluding item i). The two coefficients β and η are defined in equation (4). The quantity $\overline{m}_z^{-i}$ represents the sum of $\overline{m}_{z,t}^{-i}$ over all possible term t, which serves as a normalizer in equality (12). Similarly, if z is a new node, then

$$P(D_i \,|\, p_{i,k+1} = z, \text{rest}) = \begin{cases} \prod_{t \in D_i} \frac{\eta+1}{|V|+|D_i|} & y \text{ is the root node.} \\ \prod_{t \in D_i} \frac{\eta \cdot m_{y,t}+1}{\eta \cdot m_y+|D_i|} & \text{otherwise} \end{cases} \quad (14)$$

where V is the vocabulary that contains all possible terms.

The complexity of this sampling procedure is $O(LC)$ for each item, where L is the depth of the tree and C is the average number of children per node. Comparing to the naive implementation, the approximate sampling procedure is exponentially faster for a balanced tree.

4.2 Estimating Model Parameters

Given the tree structure, we estimate the latent factors and the popularity measures for categories and items. The parameter estimation model is based on BPR and the optimization is implemented via stochastic gradient ascent.

At each iteration, we sample a user u, a purchased item i and an unpurchased item j. Suppose that $i_0 \to i_1 \to \cdots i_{L_i} = i$ represents the path from the root node to i. We define the same notation for item j. Given the conditional probability (7), the log-posterior of the observation that user u prefers item i over item j is given by

$$L_{uij} = \log \sigma(x_{ui} - x_{uj}) - \sum_{k=0}^{L_i-1} \frac{\|v_{i_k} - v_{i_{k+1}}\|^2}{2\sigma^2} - \frac{q_i^2}{2\tau^2} - \sum_{k=1}^{L_j} \frac{\|v_{j_k} - v_{j_{k-1}}\|^2}{2\sigma^2} - \frac{q_j^2}{2\tau^2},$$

We update the parameters in this local objective function by stochastic gradient ascent. The first step is to compute the local derivatives. Let c_{uij} denote the quantity $1-\sigma(x_{ui}-x_{uj})$. Then we find that

$$\frac{\partial L_{uij}}{\partial v_u} = c_{uij}(v_i - v_j) - \frac{v_u}{\sigma^2}. \quad (15)$$

For latent factors, we update the difference between every latent factor v_{i_k} to its parent $v_{i_{k-1}}$. Then all latent factors are implicitly updated once their differences to the parent are updated. Let $w_{i_k} = v_{i_k} - v_{i_{k-1}}$ be such a shorthand notation, then we have

$$\frac{\partial L_{uij}}{\partial w_{i_k}} = c_{uij} v_u - \frac{v_{i_k}}{\sigma^2} \quad k = 1, \ldots, L_i; \quad (16)$$

$$\frac{\partial L_{uij}}{\partial w_{j_k}} = -c_{uij} v_u - \frac{v_{j_k}}{\sigma^2} \quad k = 1, \ldots, L_j. \quad (17)$$

For popularity measures q_i and q_j, since they are part of a non-smooth operator $[\cdot]^+$, we first approximate the operator by a differentiable function

$$g(x) = \delta \log(1 + \exp(x/\delta))$$

Table 2: Metadata used in experiments.

description	Description of the item (string)
brand	Brand of the item (string)
price	Discretized price of the item

for some small constant δ and then compute the partial derivatives by

$$\frac{\partial L_{uij}}{\partial q_i} = \frac{c_{uij} e^{q_i/\delta}}{1 + e^{q_i/\delta}} - \frac{q_i}{\tau^2}, \tag{18}$$

$$\frac{\partial L_{uij}}{\partial q_j} = -\frac{c_{uij} e^{q_j/\delta}}{1 + e^{q_j/\delta}} - \frac{q_j}{\tau^2}. \tag{19}$$

According to equations (15)-(19), we update the parameters $\theta \in \{v_u, w_{i_k}, w_{j_k}, q_i, q_j\}$ via stochastic gradient ascent:

$$\theta \leftarrow \theta + \epsilon \frac{\partial L_{uij}}{\partial \theta}. \tag{20}$$

where ϵ is the stepsize. The update terminates when all parameters of the model converge.

5. EXPERIMENTS

In this section, we present experimental evaluations for the hierarchical latent factor model. We compare the HF model with three state-of-the-art latent factor models: the classical latent factor model (MF), the collaborative item selection model (CIS) and the regression-based latent factor model (RLFM).

5.1 Dataset

We used a log of user online transactions obtained from a major search engine, email provider and online shopping site. The dataset contains information about the historical purchases of users over a period of 3 months. We fully anonymize the users by dropping the original user identifier and assigning a new, sequential numbering of the records. As a result, we have about 14 million anonymized users with an average of 2.4 purchases per user and 3.28 million individual products.

We also group items with respect to their frequencies (number of purchases in the log data) and summarize the result in Figure 2. As Figure 2(a) shows, the majority of all items have frequency of at most 10. However, as Figure 2(b) shows, a majority of all purchases occur with high-frequency items (item of frequency greater than 100). In other words, most of the items in the dataset don't have sufficient purchases for estimating their parameters. This unbalanced distribution of data characterizes the challenge of our task.

We partition the purchase history of each user into two parts: training and testing. The first part contains 1/2 of the purchased items and the second part contains the remaining 1/2. Then, we take the first part as the training data and second part as the test data. In particular, if the user bought only one item, then we assign it to the training set. This results in about 18.6 million purchases for training and about 14.8 million purchases for testing.

If we examine the data distribution in Figure 2(a) and Figure 2(b), we find that purchases on the top 2% most frequent items actually occupy more than 60% of the overall purchases. This unbalanced data distribution makes the gap between different approaches small, because even for the simplest latent factor model (such as MF), as long as it achieves good performance on the top-frequency items, it achieves good overall performance. We construct a sparse data set that tests the models' capability of learning from all items. In particular, we remove those "trivial" users that only buy popular items and keep those "non-trivial" users that have bought at least one tail item (item with frequency $1-10$). As shown in Figure 2(c), it makes training and recommendation more challenging since infrequent items have more weight in the sparse dataset. By this construction, we obtain 3.5 million users and 12.4 million purchases. We employ the same strategy as in the previous paragraph to partition the training set and the test set which gives us 6.1 million purchases for training and 6.3 million purchases for testing. The sparse dataset contains about 37% users in the original full dataset.

In our implementation, every user and every item is represented by a unique id. The item id and user id are used by all models to construct latent factors. Besides using purchase data, we leverage three types of meta-information to help recommendation. These meta-information, which we summarize in Table 2, are used by the RLFM model and the HF model. In particular, the RLFM model uses all three types of meta-information. The HF model uses the item description information. Note that it is possible to modify the HF model's specification in Section 3 to allow it accepting brand and price features. We don't do it in this paper since we want the model specification to be concise as possible and only using the description information turns out to achieve good performance.

5.2 Implementation Details

We developed a multi-core implementation of all five models in C++. In latent factor models, we choose the factor dimensions $d \in \{10, 20, 40, 60, 80\}$ where $d = 20$ is the default setting. The latent factors are initialized by multivariate Gaussian $\mathcal{N}(0, 0.1 \times \mathbf{1})$. For stochastic gradient descent, we control the gradient stepsize using the adaptive gradient method [5]. For all experiments, we use regularization coefficients obtained by cross validation. The constant δ that approximates the operator $[\cdot]^+$ is set to be 0.2.

5.3 Evaluation

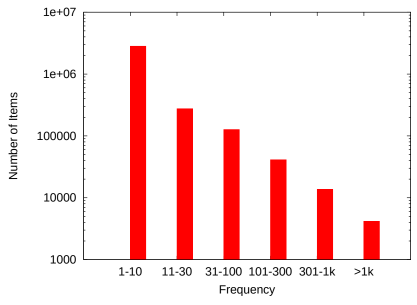
We use the AUC (Area under the ROC curve) metric to compare the performance of models. AUC is a widely used metric for testing recommender systems and latent factor models [14, 1, 7]. Let X be the set of all products. Given a user u, we suppose that $r(u,i)$ is the numerical rank of item $i \in X$ provided by some model $\mathcal{M}$. Let T_u be the set of items the user u purchased in the test set, the formula to compute AUC is given by: (Here $\delta(x)$ is the indicator function that returns 1 if x is true or 0 otherwise):

$$\text{AUC}_u = \frac{1}{|T_u||X \backslash T_u|} \sum_{i \in T_u,\ j \in I \backslash T_u} \delta(r(u,i) < r(u,j))$$

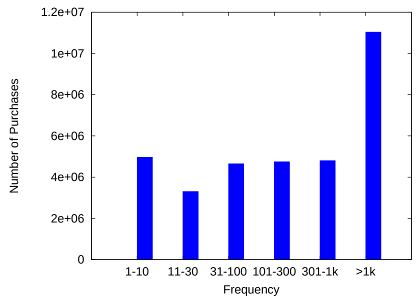
The AUC on the overall test set is the average individual user's AUC weighted by the size of their purchases, that is

$$\text{AUC} = \frac{\sum_u |T_u| \text{AUC}_u}{\sum_u |T_u|}$$

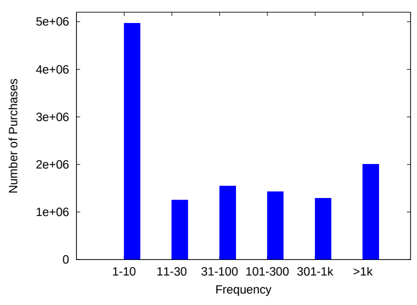
It is straightforward to see that AUC is a value in the range of $[0, 1]$. A greater AUC indicates a better ranking quality provided by the model $\mathcal{M}$.



(a) Item distribution



(b) Purchase distribution on full data

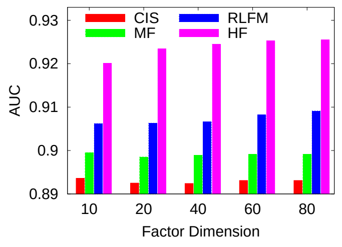


(c) Purchase distribution on sparse data

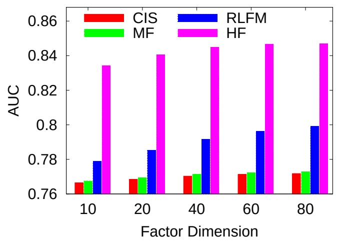
Figure 2: The figures show the distribution of items and purchases in the dataset. (a) Distribution of items in each frequency group. (b) Distribution of purchases in each frequency group. (c) Distribution of purchases in the sparse dataset. For the sparse dataset, we keep users that have bought at least one infrequent item.

Table 3: Comparing models on the full set with AUC metric. Bold numbers indicate the best performance and the star indicates statistical significance (p-value < 0.01).

Item Frequency	1 - 10	11 - 30	31 - 100	101 - 300	301 - 1000	$>$ 1000	Overall
MF	0.453	0.878	0.961	0.987	0.996	0.9996	0.899
CIS	0.444	0.860	0.948	0.982	0.995	0.9996	0.893
RLFM	0.529	0.863	0.957	0.987	0.996	0.9995	0.908
HF	**0.617***	**0.891***	**0.965***	**0.989**	**0.997**	**0.9996**	**0.925***



(a) On full data



(b) On sparse data

Figure 3: Comparing latent factor model performances with factor dimensions $d \in \{10, 20, 40, 60, 80\}$. The HF model outperforms the MF, CIS and RLFM models.

5.4 Comparing Latent Factor Models

In this section, we compare the performance of latent factor models on full dataset and sparse dataset. The evaluation results are summarized in Figure 3(a) and Figure 3(b).

Among the four models, the MF model and the CIS model have similar performances, and the RLFM model is slightly better. The HF model yields significantly better performance than the three baseline methods. Note that the HF model uses only a subset of meta-information that is used by the RLFM model. It suggests that the hierarchical structure in the HF model organizes the meta-information in a more efficient way than in the RLFM model's regression approach.

In order to examine more carefully the evaluation results, we pay more attention to the setting of $d = 60$ (best results for most models) and we partition the test set into six frequency groups to evaluate performances on each individual group. The results are reported in Table 3 and Table 4. From these tables, we find that the biggest performance gaps are among the low-frequency groups. For items that have frequency of $1-10$, the RLFM model, which leverages meta-information, is better than the MF model and the CIS model that only use user purchase data. Furthermore, the HF model which maintain structural categorization of items is much better than the RLFM model. It confirms that hierarchical categorization is especially helpful to low-frequency items.

Finally, as plots and tables suggest, the HF model's improvement is more significant on the sparse dataset. One reason is that the sparse dataset is harder for training since the user-item interaction is insufficient. The HF model, which allows sharing purchase data among infrequent items under hierarchical categorization, is more robust to sparsity. This intuition is confirmed by Table 3 and Table 4's column for frequency $1-10$, where the performance gap between HF and RLFM is greater on the sparse data (12% for sparse data versus 9% for full data). On the other hand, the sparse dataset is also more challenging for testing since the low-frequency items occupy more weight as suggested by Figure 2(c). This makes the HF model's improvement over low-frequency items to be more noticeably reflected in the overall performance. As a comparison, the HF model is at least 1.7% better than the baseline models on the full dataset, and at least 5.1% better on the sparse dataset.

5.5 An Example of Algorithm-generated Taxonomy

In Figure 4, we present a portion of the hierarchical categories discovered by the HF model. Besides the top-ranked terms in each category, we also manually labeled the category names to make them more readable. As Figure 4 shows, the HF model is capable of constructing a high-quality hierarchical structure of categories. Categories in higher level represent broader concepts, and their sub-categories represents more refined range of products. For example, the taxonomy in Figure 4 clustered clothing items together, and refines the category by jeans, dresses and polos. It also divide the jeans category into two smaller sub-categories separating men's jeans and women's jeans, which is in analogy to the taxonomy that is created by human. Note that the hierarchical tree used by the HF model is automatically generated and dynamically updated. Thus, unlike the static

Table 4: Comparing models on the sparse dataset with AUC metric. For this dataset, we keep users that have bought at least one infrequent item. Bold symbols indicate best performance and the star indicates the statistical significance (p-value < 0.01).

Item Frequency	1 - 10	11 - 30	31 - 100	101 - 300	301 - 1000	> 1000	Overall
MF	0.479	0.707	0.913	**0.983**	**0.995**	**0.9993**	0.772
CIS	0.472	0.720	0.916	0.978	0.993	0.9993	0.771
RLFM	0.544	0.741	0.904	0.976	0.994	0.9992	0.796
HF	**0.662***	**0.798***	**0.923***	0.981	0.994	0.9992	**0.847***

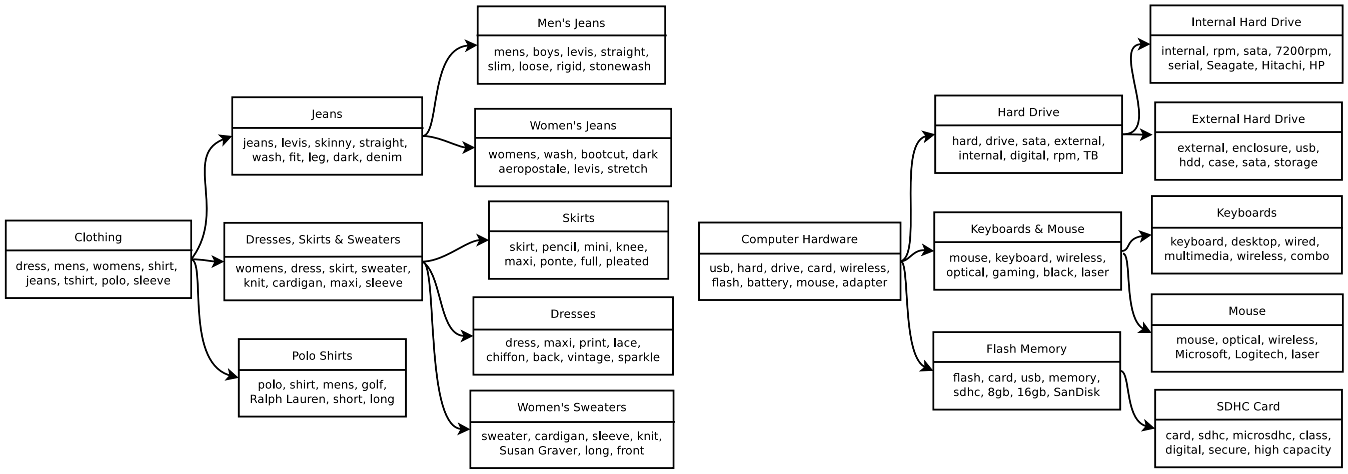

Clothing
dress, mens, womens, shirt, jeans, tshirt, polo, sleeve
Jeans
jeans, levis, skinny, straight, wash, fit, leg, dark, denim
Men's Jeans
mens, boys, levis, straight, slim, loose, rigid, stonewash
Women's Jeans
womens, wash, bootcut, dark aeropostale, levis, stretch
Dresses, Skirts & Sweaters
womens, dress, skirt, sweater, knit, cardigan, maxi, sleeve
Skirts
skirt, pencil, mini, knee, maxi, ponte, full, pleated
Dresses
dress, maxi, print, lace, chiffon, back, vintage, sparkle
Women's Sweaters
sweater, cardigan, sleeve, knit, Susan Graver, long, front
Polo Shirts
polo, shirt, mens, golf, Ralph Lauren, short, long
Computer Hardware
usb, hard, drive, card, wireless, flash, battery, mouse, adapter
Hard Drive
hard, drive, sata, external, internal, digital, rpm, TB
Internal Hard Drive
internal, rpm, sata, 7200rpm, serial, Seagate, Hitachi, HP
External Hard Drive
external, enclosure, usb, hdd, case, sata, storage
Keyboards & Mouse
mouse, keyboard, wireless, optical, gaming, black, laser
Keyboards
keyboard, desktop, wired, multimedia, wireless, combo
Mouse
mouse, optical, wireless, Microsoft, Logitech, laser
Flash Memory
flash, card, usb, memory, sdhc, 8gb, 16gb, SanDisk
SDHC Card
card, sdhc, microsdhc, class, digital, secure, high capacity

Figure 4: A portion of hierarchical categories discovered by the HF model. The diagram shows two categories and a subset of their sub-categories. Each block shows the top ranked terms in the category and a manual labeling of the category's name based on the terms.

human-induced taxonomy, the algorithm-generated taxonomy is adaptive to incremental data as more items and more users arrive.

6. USING HUMAN-INDUCED TAXONOMY

In this section, we assume that the human-induced taxonomy is available to the recommendation system. We study approaches that incorporate the human-induced taxonomy into the HF model. We also compare the HF model with the taxonomy-aware latent factor model (TF), which is the state-of-the-art latent factor model based on human-induced taxonomies. Even without using the human-induced taxonomy, we find that the HF model consistently outperforms the TF model. Incorporating human-induced taxonomy further improves the HF model's accuracy to make it achieving the best performance in our comparison.

We begin with describing the human-induced taxonomy that we use in this section. In this reference taxonomy, products are organized by a tree-structured taxonomy that has 20 top-level categories and 5140 internal nodes. The average depth of the tree is 4.4. Each items is attached to exactly one internal node of the tree.

6.1 Including Human-induced Taxonomy in Generative Model

When a human-induced taxonomy is available to the HF model, we can include it as a part of the HF model. More specifically, we assume that these taxonomies are also generated by the underlying hierarchical model described in Section 3. Then, by observing the human-induced taxonomy as long as the item descriptions and user purchase data, we use the technique in Section 4 to infer the model structure and the model parameters.

Let A_i be the set of ancestors of item i in the human-induced taxonomy tree. A_i can be seen as a collection of terms, which provides another description to the item. For every category node in the HF model, we maintain another multinomial distribution that is used for generating such descriptions. When a new category node w is generated, we sample the multinomial distribution by

$$\varphi_w \sim \begin{cases} \mathrm{Dir}(\beta') & w \text{ is the root node} \\ \mathrm{Dir}(\eta' \varphi_{\pi(w)}) & \text{otherwise} \end{cases}$$

and when an item i is generated, we sample each term a in the associated set A_i by the multinomial distribution

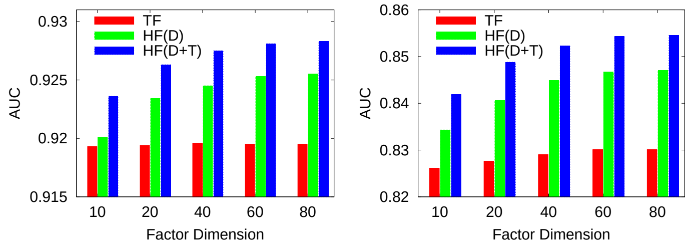
$$a \sim \mathrm{Mult}(\varphi_{\pi(i)}).$$

By setting hyper-parameters β' and η', we can control the weight of the human-induced taxonomy in the global objective function. A smaller value of β' or η' indicates that the human-induced taxonomy is more strictly followed when the hierarchical structure is constructed. In practice, we set the values of β' or η' using cross validation.

The inference for this modified HF model still follows the approximation algorithm described in Section 4. In equation (9), we multiply an additional term describing the likelihood of the observed description A_i, whose computation is in complete analoge to the likelihood term for D_i using equations (12-14). We omit the mathematical details here due to space limitations.

Table 5: Number of internal nodes, entropies and their mutual information for three categorizations. In this table, $\mathrm{T}_{\mathrm{human}}$ is the taxonomy created by human, $\mathrm{T}_{\mathrm{HF(D)}}$ and $\mathrm{T}_{\mathrm{HF(D+T)}}$ are taxonomies generated by the HF(D) model and the HF(D+T) model.

Categorization	# Nodes	Entropy	Mutual Information
$\mathrm{T}_{\mathrm{human}}$	5140	5.976	-
$\mathrm{T}_{\mathrm{HF(D)}}$	7424	8.247	-
$\mathrm{T}_{\mathrm{HF(D+T)}}$	7356	8.182	-
$\mathrm{T}_{\mathrm{human}}$, $\mathrm{T}_{\mathrm{HF(D)}}$	-	-	3.957
$\mathrm{T}_{\mathrm{human}}$, $\mathrm{T}_{\mathrm{HF(D+T)}}$	-	-	5.162



(a) On full data (b) On sparse data

Figure 5: Using only item descriptions, the HF(D) model outperforms the TF model which relies on human-induced taxonomy. The HF(D+T) model achieves the best performance.

We rename the two variants of the HF model by HF(D) and HF(D+T), indicating that the model relies on only the item description or relies on both the description and the human-induced taxonomy. In next section, we compare the taxonomy generated by HF(D) or HF(D+T) to the taxonomy induced by human.

6.2 Comparing Taxonomies induced by Human and by Algorithm

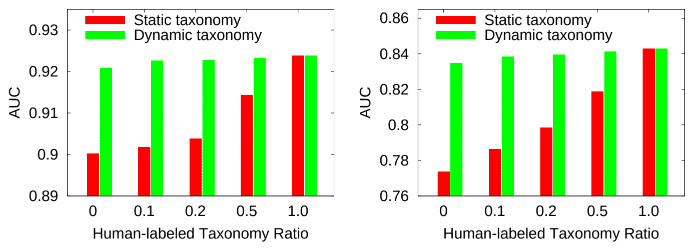
To measure the connection between taxonomies that the HF model generates and the taxonomy induced by human, we compute the entropy of each taxonomy and their mutual information. Let C be the set of categories in taxonomy T. Let p_c be the portion of items belonging to category $c \in C$. Then, the entropy of T is defined by

$$H(T) = \sum_{c \in C} -p_c \log(p_c).$$

For two taxonomies T_1 and T_2, their joint entropy is calculated based on the Cartesian product of their category set $C_1 \otimes C_2$, and their mutual information is defined by

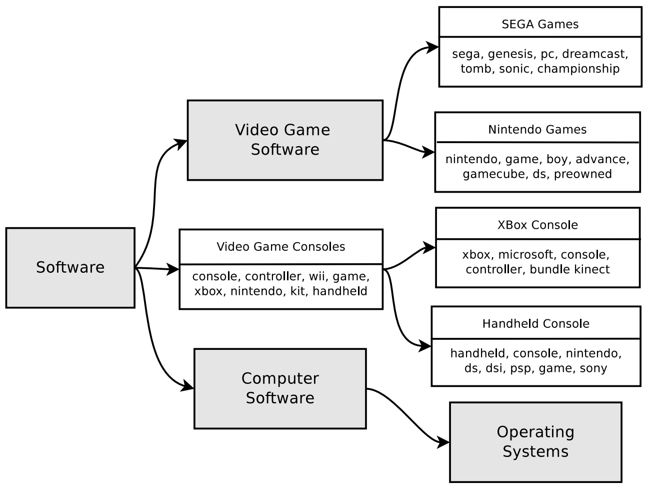
$$I(T_1, T_2) = H(T_1) + H(T_2) - H(T_1, T_2).$$

A high mutual information indicates that two taxonomies are strongly correlated. As Table 5 shows, taxonomies generated by the HF model have the similar number of internal nodes and entropies as the human-induced taxonomy. The HF(D) model is capable of producing categories that has high mutual information with the human-induced taxonomy (equals 3.957). When human-induced taxonomy is incorporated to form the HF(D+T) model, the mutual information increases to 5.162, which means that the resulting taxonomy preserves most of the information induced by human.



(a) On full data (b) On sparse data

Figure 6: When human-induced taxonomy is partially available, the dynamic taxonomy generated by the HF model leads to robust recommendation performance.



Software
Video Game Software
Video Game Consoles
console, controller, wii, game, xbox, nintendo, kit, handheld
Computer Software
SEGA Games
sega, genesis, pc, dreamcast, tomb, sonic, championship
Nintendo Games
nintendo, game, boy, advance, gamecube, ds, preowned
XBox Console
xbox, microsoft, console, controller, bundle kinect
Handheld Console
handheld, console, nintendo, ds, dsi, psp, game, sony
Operating Systems

Figure 7: A portion of dynamic taxonomy generated by the HF model for $r = 0.1$. The shaded box represents human-created categories. The white box represents automatically generated categories with top-ranked terms and manually labeled category names.

Next, we compare the three taxonomies in recommender systems. As Figure 5(a) and Figure 5(b) shows, the HF(D) models, though only relying on the raw data, consistently outperforms the TF model which relies on human labels. The HF(D+T) model outperforms all other models on both datasets. The HF(D+T) model's improvement over the TF model is 0.9% on the full data and 2.4% on the sparse data, both are statistically significant. The experiment results confirm our intuition that the HF model has advantage over the TF model in recommendation accuracy, since it combines categorization and parameter estimation together and optimize both jointly. In contrast, the human-induced taxonomy used by the TF model is not optimized for learning latent factor models.

6.3 Dynamically Evolving Human-induced Taxonomy

In this section, we consider a scenario when the human-induced taxonomy is taken as the ground truth, but it is not fully available. That means, some of the items are categorized by human and others remains uncategorized. This commonly happens when new source of items are added to the database or when new types of products appear in the market. In this case, the recommender doesn't want to completely reconstruct the taxonomy that is already labeled by editors. Instead, he/she desires a dynamic hierarchy: old items keep their positions in the existing taxonomy, new items are automatically added to the taxonomy, and new categories are inserted when necessary.

The HF model is capable of maintaining such a dynamic taxonomy. In particular, In section 4.1 we initialize the tree with the human-induced taxonomy so that existing items always belong to their human-induced categories. When new items arrive, they are assigned to existing categories or assigned to new categories according to the sampling algorithm in Section 4.1.

We simulate a partial human-induced taxonomy in experiment by taking a ratio $r \in \{0, 0.1, 0.2, 0.5, 1\}$ of items categorized by the human-induced taxonomy, and keeping the remaining items uncategorized. We compare the HF model with static tree where uncategorized items are directly attached to the root, and the HF model with the dynamic tree described above. According to the experiment results, the dynamic taxonomy appears to be very robust to incomplete categorization. As Figure 6 shows, when using a static tree, the latent factor model's performance dramatically decreases as the ratio r goes down, but with the dynamic tree, the performance always retains at a high level. It suggests that in reality the human editor only need to label a small portion of items, then the algorithm will complete the remaining part.

In Figure 7, we present a portion of the dynamic tree to illustrate how the taxonomy envolves. In this example, the HF model adds sub-categories to the "Video Game Software" node to further refine the categorization. Interestingly, it also creates a "Video Game Consoles" category that does not belong to the original taxonomy. Although the game consoles are literally not software, they are indeed closely related to the game software purchase, which makes the resulting taxonomy a reasonable priori for recommendation.

7. CONCLUSIONS

In this paper we addressed the problem of inferring a taxonomy for recommender systems. Smoothing latent factor models over a taxonomy combats sparsity and allows for sharing statistical strength between items. However, in many situations a human-induced taxonomy is not available, for example when new items arrive, when a new merchant is added to the system (possibly with items from a different culture/language), or when users do not supply categories for new items (as in youtube videos). Luckily, it is always possible to obtain a textual description of items. We described an unsupervised non-parametric method that jointly learns taxonomy structure over items and item factors in a recommender system from both items' textual description and purchase data. We showed that the performance of our model compares favourably with several state of the art baselines and even with the performance of a human-induced taxonomy when available. Furthermore, we showed that our model can utilize and improve upon a partial human-induced taxonomy if available. In the future we plan to apply our taxonomy induction to the regression latent factor model in [1]. Moreover, we plan to investigate user clustering when users' meta data is available.

8. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 19–28. ACM, 2009.

[2] A. Ahmed, L. Hong, and A. Smola. The nested chinese restaurant franchise process: User tracking and document modeling. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[4] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):7, 2010.

[5] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2010.

[6] Z. Ghahramani, M. I. Jordan, and R. P. Adams. Tree-structured stick breaking for hierarchical data. In *Advances in Neural Information Processing Systems*, pages 19–27, 2010.

[7] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan, and L. Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *Proceedings of the VLDB Endowment*, 5(10):956–967, 2012.

[8] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434. ACM, 2008.

[9] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[11] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 141–149. ACM, 2011.

[12] A. Mnih. Taxonomy-informed latent factor models for implicit feedback. 2011.

[13] A. Mnih and Y. W. Teh. Learning label trees for probabilistic modelling of implicit feedback. In *Advances in Neural Information Processing Systems*, pages 2825–2833, 2012.

[14] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[15] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the 3rd ACM international Conference on Web Search and Data Mining*, pages 81–90. ACM, 2010.

[16] L.-T. Weng, Y. Xu, Y. Li, and R. Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 113–120. IEEE, 2008.

[17] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 406–415. ACM, 2004.