Modeling of Multilayer Multicontent Latent Tree and Its Applications

Chia-Yu Lin, Yu-Fang Chiu, Li-Chun Wang, *Fellow, IEEE*, and Dusit Niyato, *Fellow, IEEE*

***Abstract*— Latent tree model (LTM) is a probabilistic tree-structured graphical model, which can reveal the hidden hierarchical causal relations among data contents and play a key role in explainable artificial intelligence. However, because current LTM modeling techniques are only suitable for single-content variable, the applications of LTMs are somewhat limited. Toward this end, a multilayer LTM (ML-LTM) is first presented to deal with the hierarchical clustering issues of multicontent variables. Second, we further develop an ML-LTM-based multicontent recommendation system. Our experiment results show that the proposed ML-LTM can achieve 90% recommendation accuracy, but the current LTM can only has 20%. Third, we propose an incremental update approach for ML-LTM that can save five-sixth updating time comparing with the whole-model retraining approach for achieving the same recommendation accuracy.**

***Index Terms*— Incremental update techniques, latent tree model (LTM), multicontent data, recommendation systems.**

I. Introduction

DISCOVERING latent variables to explain their latent dependencies are the key in explainable artificial intelligence for data science. The explanation of latent variables helps users make better decisions and increase their acceptance of suggestions [1]–[3]. In general, there are two types of discovering latent variables approaches: 1) matrix factorization and 2) latent tree model (LTM). Matrix factorization characterizes data along each dimension by latent vectors and calculates the similarity between users and items in the latent space through inner product [4]. However, matrix factorization can only show the similarity between variables, instead of explaining the meanings of latent variables [5]. LTM utilizes a probabilistic tree to explain hidden complex relationships among data [6]. Fig. 1 is an example of a latent tree over cloth data. Twill, diamond, hound's tooth, stripe, and check are five observed variables of pattern. "Yes" and "No" are two states of observed variables to indicate the presence or absence of variables. In LTM, latent variable L_1 indicates the twill pattern of "twill, diamond, and hound's tooth." L_2 clusters the strip pattern of "strip and check." Considering three observed variables (twill, diamond, and hound's tooth), there are eight possible states. When we observe the data, we find that data are gathered in four states. These four states are called latent states to represent the meaning of latent variables. The number of latent states is shown in parentheses. Latent variables and latent states represent the latent relationship among observed variables. LTM has recently received widespread attention, since the dependencies in complex data sets can be clearly explained by the tree structure with linear complexity [7], [8].

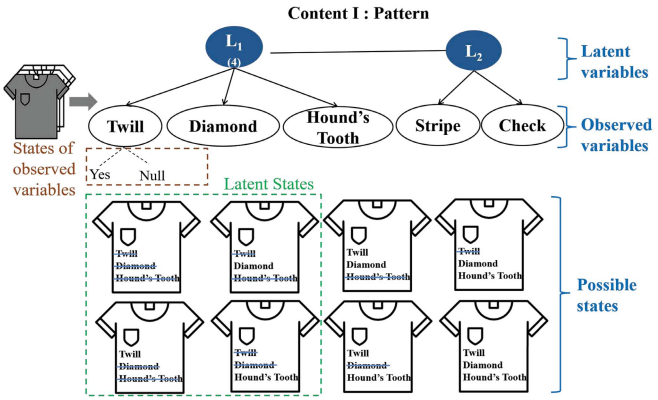

Fig. 1. Example of latent tree based on cloth data.

However, most existing LTMs only consider one content variable instead of multicontent variables. Take the medical data as an example [9]–[11]. Collins and Lanza [9] adopted LTM to analyze the result of healthy behavior questionnaires. Zhang *et al.* [10] utilized LTM to find a group of syndromes and helped doctors classify patients into different classes. Zhao *et al.* [11] built LTM to discover symptom co-occurrence patterns from depressive patients. The LTMs in these works only dealt with single content data, such as syndromes. In practice, medical data contained syndromes and medicines, which should be considered together in the analysis process. To the best of our knowledge, the multicontent analysis for LTM was rarely seen in the literature.

In this article, we propose a multilayer LTM (ML-LTM) based on bridge-islands (BI) algorithm [12] for multicontent data. In the ML-LTM, individual LTM and intermediate LTM are the key roles in discovering the relevance among variables with different contents. We also utilize joint probability (JP), mutual information (MI), and conditional MI to find

Manuscript received January 6, 2020; revised September 19, 2020; accepted October 17, 2020. Date of publication November 18, 2020; date of current version January 29, 2021. This work was supported in part by the National Chiao Tung University and Ministry of Education, in part by the Ministry of Science and Technology (MOST), and in part by the Pervasive Artificial Intelligence Research Laboratories through the Higher Education Sprout Project, under Grant MOST 109-2634-F-009-018. *(Corresponding author: Li-Chun Wang.)*

Chia-Yu Lin is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan City 32003, Taiwan.

Yu-Fang Chiu and Li-Chun Wang are with the Department of Electrical and Computer Engineering, National Chiao Tung University Guangfu Campus, Hsinchu 30010, Taiwan (e-mail: lichun@g2.nctu.edu.tw).

Dusit Niyato is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798.

Digital Object Identifier 10.1109/TCSS.2020.3035202

the relevant latent variables and demonstrate ML-LTM on multicontent recommendation systems.

Another challenge of LTM is the accuracy issue when new data come into the system. The original LTM built in the batch mode may become obsolete and result in low accuracy. Existing works focus on improving the efficiency of building latent tree with many observed variables instead of solving the issue of model updating [12], [13]. The computation cost of retraining LTM is high. Updating the LTM model in real time is a big challenge.

An incremental update mechanism for efficiently updating ML-LTM is proposed in this article. We distinguish unstable variables by MI and information coverage. During the updating process, we only update the unstable variables to reduce the computing complexity of model updating.

The contribution of this article is summarized as follows.

1) For multicontent data, the proposed approach is the first ML-LTM in the literature.
2) From the experiments of multicontent recommendation system, the proposed ML-LTM can achieve 90% accuracy, while the traditional LTM can only get 20% accuracy.
3) The proposed incremental update approach can reach equivalent accuracy, but only requires one-sixth update time compared with the whole-model retraining approach. As mentioned earlier, ML-LTM is an efficient model for discovering latent variables among multicontent data in many applications.

The rest of this article is organized as follows. Section II introduces the background and related works of LTM. Section III presents the ML-LTM for multicontent variables. Section IV deploys the ML-LTM for a recommendation system. The experiments and numerical results are shown in Section V. Finally, we give our concluding remarks in Section VI.

II. Background and Related Work

In this section, we introduce the basic concepts of LTM and the works of LTMs in the recommendation system and model updating.

A. Latent Tree Model Construction

LTM utilized latent variables to find relationships among observed variables and connect them through a tree structure. LTM was originated from the Bayesian network, which was regarded as a directed probabilistic graphical model. In LTM, leaf nodes were observed variables and intermediate nodes were latent variables. Given an observed variable O_i with latent variable L_i, the probabilistic dependence between two nodes could be defined by a conditional distribution $P(O_i|L_i)$. In general, assume there were n observed variables $(O) = O_1, O_2, \ldots, O_n$ and k latent variables $(L) = L_1, L_2, \ldots, L_k$. $pa(X)$ denoted the parent of a variable X. When X was the root, let $pa(X)$ be an empty set. The joint distribution over observed and latent variables by LTM was

$$\mathrm{P(O_1,\ldots,O_n,L_1,\ldots,L_k)} = \prod_{\mathrm{X\in O\cup L}} P(X|pa(X)). \quad (1)$$

The goal of building an LTM m was maximizing Bayesian information criterion (BIC) score was used for scoring LTMs [14], as follows:

$$\mathrm{BIC(m|D)} = \log \mathrm{P(D|m,}\theta^*) - \frac{d(m)}{2}\log N \quad (2)$$

where θ^* was the maximum likelihood estimation of the parameters. $d(m)$ was the number of free probability parameters in m. N was the sample size. A variety of learning LTM methods were proposed previously [6], [13], [15], [16]. Zhang [6] proposed a hierarchical latent class model (LCM) for cluster analysis. In the experiments of [6], they only considered data with 16 attributes. Zhang and Kocka [15] designed two computationally, efficient algorithms for learning minimal latent trees. They simulated algorithms based on 80 observed variables. Choi *et al.* [13] presented a method for performing multidimensional clustering on categorical data and showed its superiority over unidimensional clustering. In the experiments, we adopted two data sets with four binary manifest variables. These works could not handle data with hundreds or more attributes. Liu *et al.* [12] proposed a BI algorithm to build LTM with hundreds of attributes. There were three steps of BI algorithm for learning LTM: 1) group similar content variables and find latent variables; 2) connect subtrees by the Chow–Liu algorithm [17]; and 3) refine the tree based on the global view by expectation–maximum (EM) algorithm [18].

The objective of the first step was determining sibling clusters. To identify potential siblings, MI [19] was adopted to decide the mutual dependence between two variables, as follows:

$$\mathrm{I(X;}\, Y) = \sum_{X,Y} P(X,Y)\log\frac{P(X,Y)}{P(X)P(Y)}. \quad (3)$$

The pair of variables with the highest MI initially formed the sibling cluster. The new variable was added to the sibling cluster if it had the highest MI with the current cluster. The MI of a new variable X and a sibling cluster C was calculated as follows:

$$\mathrm{I(X;}\, C) = \max_{Z\in C} \sum_{X,Z} P(X,Z)\log\frac{P(X,Z)}{P(X)P(Z)}. \quad (4)$$

In the second step, the Chow–Liu algorithm connected subtrees. The inputs of the Chow–Liu algorithm were latent variables. The latent variables with the highest MI were connected. The connecting process was repeated until all latent variables were connected to generate a latent tree.

EM-algorithm refined LTM based on global consideration in the final step. The probability of latent classes and the conditional probability between observed and latent classes were optimized. EM algorithm was repeated until the convergence was found.

B. Latent Tree Model for Recommendation Systems

LTM was utilized to explain the clusters of variables. Take the medical data as an example. Yang *et al.* [20] identified the symptoms in psoriatic patients based on latent class analysis (LCA). In this article, 507 psoriatic patients were

clustered into three symptoms by LCA. Zhang *et al.* [10] constructed the relation between syndromes and the diseases and classified patients into different classes by LTM. Zhao *et al.* [11] adopted LTM to discover symptom co-occurrence patterns from 604 cases of depressive patients. An LTM with 29 latent variables was built in [11]. Probabilistic symptom co-occurrence patterns were captured by some latent variables. However, in previous studies, doctors could only group patients by their syndromes but could not group the relationship between symptoms and medicines through LTM. The recommendation among multicontent variables based on LTM is essential and has not been mentioned in the literature.

C. Updating Latent Tree Model

There were many works focused on the building of LTM. Liu *et al.* [12] built LTM with hundreds of attributes by proposing a BI algorithm. Choi *et al.* [13] developed recursive grouping and CLGrouping for the learning of latent tree based on a distance-based framework. Liu *et al.* [21] proposed a hierarchical latent tree analysis (HLTA) to determine a topic of documents. Balakrishnan and Chopra [22] adopted the method of gradient acceleration optimization to improve the execution efficiency of progressive EM HLTA (PEM-HLTA) for topic detection. References [12], [13], [21], and [22] only considered how to build LTM with a large amount of data. However, in current applications, new data frequently enter the system [23]. Traditional LTM was built in the batch mode. When new data came in, the LTM might become obsolete and result in low accuracy. Lin *et al.* [4] discussed the accuracy issue of recommendation systems when the speed of the new input data increased. Efficiently updating the LTM model based on new data becomes important.

III. Multilayer Latent Tree Model for Multicontent Variables

We propose an ML-LTM for multicontent data. Since a traditional latent tree cannot represent the relationship of multicontent variables, we design an intermediate tree in ML-LTM to connect different content data. ML-LTM consists of three key elements: 1) individual LTM; 2) intermediate LTM; and 3) possible states. We define the parameters of ML-LTM, as shown in Table I. The input data of the ML-LTM contains N contents. In LTM, leaf nodes are observed variables. $O_1^n, O_2^n, \ldots, O_i^n$ denotes the observed variable of content n. $u_{i,1}^n, u_{i,2}^n, \ldots, u_{i,j}^n$ are the states of the observed variable O_i^n. In LTM, intermediate nodes are latent variables. $L_1^n, L_2^n, \ldots, L_k^n$ denotes the latent variables of content n. $v_{k,1}^n, v_{k,2}^n, \ldots, v_{k,m}^n$ represent the latent states of the latent variable L_k^n. Given an observed variable O_i^n in the state $u_{i,j}^n$ belongs to the latent state $v_{k,m}^n$ of latent variable L_k^n, the probabilistic dependence between two nodes can be interpreted by a conditional distribution $P(O_i^n = u_{i,j}^n | L_k^n = v_{k,m}^n)$.

A. Individual LTM

We build individual LTMs by BI algorithm [12] to separate different content variables. We take two content input data as an example. There are five observed variables in content 1 and four observed variables in content 2. Table II is an example of the input data. The individual LTMs of content 1 and conten 2 are constructed, as shown in Fig. 2(a) and (b). In the LTM of content 1, observed variables O_1^1, O_2^1, and O_3^1 belong to latent variable L_1^1. In the LTM of content 2, observed variables O_1^2 and O_2^2 belong to latent variable L_1^2. Besides the clustering result of observed variables, we can obtain latent states from LTM. In Fig. 2(a) and (b), we assume that $v_{1,1}^1$ and $v_{1,2}^1$ are two latent states of L_1^1. L_1^2 has three latent states $v_{1,1}^2$, $v_{1,2}^2$, and $v_{1,3}^2$. LTM also records the probabilities of latent states in (5) and the probabilities of observed variables in different latent states in (6)

$$P\left(\mathrm{L}_k^n = v_{k,m}^n\right) \quad n \in \{1, \ldots, N\}, \quad k \in \{1, \ldots, K\}, \ m \in \{1, \ldots, M\} \tag{5}$$

$$P\left(O_i^n = u_{i,j}^n | \mathrm{L}_k^n = v_{k,m}^n\right) \quad n \in \{1, \ldots, N\}, \quad i \in \{1, \ldots, I\}, \ j \in \{1, \ldots, J\} \quad k \in \{1, \ldots, K\}, \quad m \in \{1, \ldots, M\} \tag{6}$$

TABLE I
Definition of Parameters in ML-LTM

Parameter	Definition
N	Number of contents in data.
$content\ n$	The n^{th} content.
I^n	The number of observed variables of the n^{th} content.
O_i^n	The i^{th} observed variable of the n^{th} content.
J_i^n	The number of states of observed variable O_i^n.
$u_{i,j}^n$	The j^{th} states of observed variable O_i^n.
K^n	The number of latent variables of the n^{th} content.
L_k^n	The k^{th} latent variable of the n^{th} content.
$SS_{L_k^n}$	The set of latent states of L_k^n.
M_k^n	The number of latent states of latent variable L_k^n.
$v_{k,m}^n$	The m^{th} latent states of latent variable L_k^n.
T_k^n	The number of observed variables belong to the latent variable L_k^n.
R	The number of latent variables in the intermediate tree.
H_r	The r^{th} latent variable in the intermediate tree.

TABLE II
Example of the Input Data

Data	O_1^1	...	O_5^1	O_1^2	...	O_4^2
$Data_1$	$u_{1,j}^1$	...	$u_{5,j}^1$	$u_{1,j}^2$	...	$u_{4,j}^2$
$Data_2$	$u_{1,j}^1$	...	$u_{5,j}^1$	$u_{1,j}^2$	...	$u_{4,j}^2$
...						

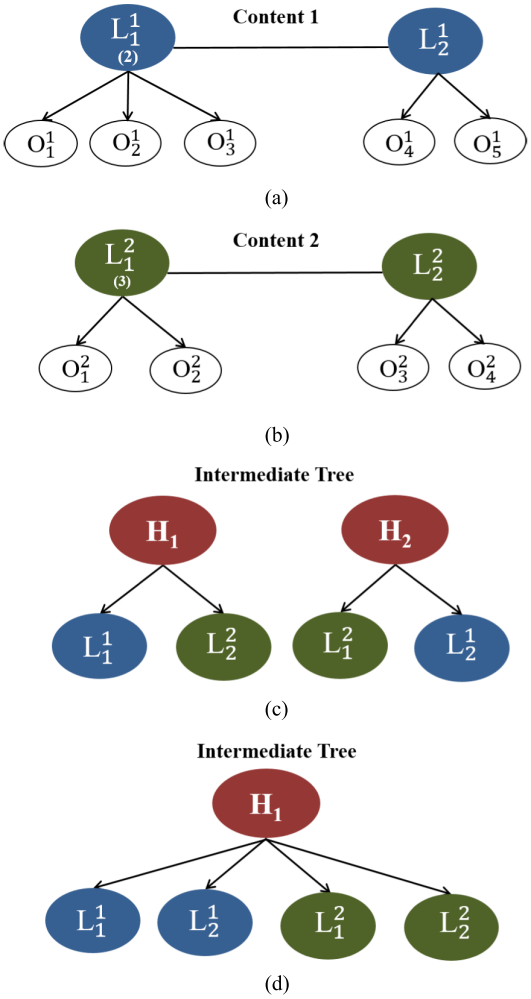

Fig. 2. Example of an ML-LTM. (a) Individual tree of content 1. (b) Individual tree of content 2. (c) Ideal intermediate tree. (d) Real intermediate tree.

B. Intermediate LTM

An intermediate latent tree is designed to connect LTMs with different content variables. Since the input data of the intermediate latent tree are latent variables and latent states, we have to transform observed variables to latent states. The transforming process is calculating the JP of observed variables and latent states. The latent state with maximum JP will be chosen to represent observed variables. We assume that T_k^n is the number of observed variables that belong to the latent variable L_k^n. M_k^n is the number of latent states of latent variable L_k^n. The objective function to find the latent state of L_k^n is shown in (7)

$$\begin{aligned}&\arg\max_{v\ \substack{m\in\{1,\ldots,M_k^n\},\\ i\in\{1,\ldots,T_k^n\}}} P\left(\mathrm{L}_k^n=v_{k,m}^n \middle| O_1^n=u_{1,j}^n,\ldots,O_i^n=u_{i,j}^n\right)\\ &=\arg\max_{v\ \substack{m\in\{1,\ldots,M_k^n\},\\ i\in\{1,\ldots,T_k^n\}}} \frac{P\left(\mathrm{L}_k^n=v_{k,m}^n,O_1^n=u_{1,j}^n,\ldots,O_i^n=u_{i,j}^n\right)}{P\left(O_1^n=u_{1,j}^n,\ldots,O_i^n=u_{i,j}^n\right)}.\end{aligned} \tag{7}$$

Since the denominator of (7) of different latent states $v_{k,1}^n, v_{k,2}^n, \ldots, v_{k,m}^n$ are the same and the variables are independent in LTM, the objective function is simplified to (8)

$$\arg\max_{v\ \substack{m\in\{1,\ldots,M_k^n\},\\ i\in\{1,\ldots,T_k^n\}}} P\left(L_k^n=v_{k,m}^n, O_1^n=u_{1,j}^n,\ldots,O_1^n=u_{i,j}^n\right) \tag{8}$$

Take Fig. 2(a) as an example, we transform Data_1 in Table II to the latent state of L_1^1 by (8). If $L_1^1 = v_{1,1}^1$ has the largest JP, $L_1^1 = v_{1,1}^1$ is chosen to represent the state ($O_1^1 = u_{1,j}^1, O_2^1 = u_{2,j}^1, O_3^1 = u_{3,j}^1$) of Data_1. The same process is executed to transform other variables. The example result of the transformation process is shown in Table III.

The intermediate tree is built based on the latent variables, and the latent states we transformed. As the definition in Table I, there are R latent variables in the intermediate tree. H_r is the rth latent variable in the intermediate tree. Fig. 2(c) is an example of an intermediate tree.

TABLE III
THE INPUT OF INTERMEDIATE TREE

Data	L_1^1	L_2^1	L_1^2	L_2^2
$Data_1$	$v_{1,1}^1$	$v_{2,2}^1$	$v_{1,2}^2$	$v_{2,1}^2$
$Data_2$	$v_{1,2}^1$	$v_{2,1}^1$	$v_{1,1}^2$	$v_{2,2}^2$
...				

C. Possible States

After building the intermediate tree, possible states are used to find the correlation among observed variables with different contents. JP is adopted to represent possible states, as shown in (9)

$$\begin{aligned}&P\left(O_1^1,\ldots,O_{t_1}^1,O_1^2,\ldots,O_{t_2}^2,\ldots,O_1^n,\ldots,O_{t_n}^n\right)\\ &\qquad 1\le t_1\le T_k^1,\quad 1\le t_2\le T_k^2,\ldots,1\le t_n\le T_k^n.\end{aligned} \tag{9}$$

Fig. 2(c) is an example of an intermediate tree, and L_1^1 and L_2^2 are connected via H_1. Therefore, O_1^1, O_2^1, O_3^1 can connect to O_3^2 and O_4^2. Thus, the possible states among O_1^1, O_2^1, O_3^1, O_3^2, and O_4^2 by JP are shown in (10)

$$\begin{aligned}&P\left(O_1^1,O_2^1,O_3^1,O_3^2,O_4^2\right)\\ &\quad=P\left(O_1^1|L_1^1\right)\times P\left(O_2^1|L_1^1\right)\times P\left(O_3^1|L_1^1\right)\times P\left(L_1^1|H_1\right)\\ &\qquad\times P\left(H_1\right)\times P\left(L_2^2|H_1\right)\times P\left(O_3^2|L_2^2\right)\times P\left(O_4^2|L_2^2\right)\\ &\quad=P\left(O_1^1|L_2^2\right)\times P\left(O_2^1|L_2^2\right)\times P\left(O_3^1|L_2^2\right)\times P\left(O_3^2|L_2^2\right)\\ &\qquad\times P\left(O_4^2|L_2^2\right).\end{aligned} \tag{10}$$

However, the number of observed variables is large in big data applications. It takes a long time to find possible states. Therefore, to decrease the complexity of possible computing states in ML-LTM, we consider the "importance" of observed variables. The importance of an observed variable is defined by the differences between the clusters after adding the observed variables. We adopt MI in (3) and information converage (IC) in (11) as the index of importance. MI represents the dependence between the observed variables and the latent variables. IC determines how much information about latent variables

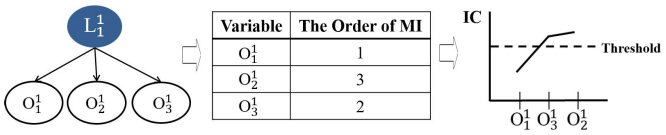

Fig. 3. The steps of considering the importance of observed variables.

can be covered by observed variables. If S_k^n represents the set of observed variables belong to latent variable L_k^n, the information coverage of the observed variable O_i^n is shown in (11). We set a threshold for IC to filter unimportant observed variables

$$\mathrm{IC}\big(O_i^n\big) = \frac{I\big(O_i^n; L_k^n\big)}{I\big(S_k^n; L_k^n\big)}. \tag{11}$$

Fig. 3 is an example of reducing the number of observed variables during computing latent classes. In the first step, we compute and sort the MI of O_1^1, O_2^1, and O_3^1. In the second step, the threshold of IC is set. Since the MI of O_1^1 is the largest, we compute the IC of O_1^1. The IC of O_1^1 is smaller than the threshold, and therefore, we compute the IC of O_1^1 and O_3^1. In Fig. 3, we can see that the IC of O_1^1 and O_3^1 is larger than the threshold. That is, O_1^1 and O_3^1 cover most of the information of L_1^1. We filter O_2^1 as an unimportant variable to decrease the computation complexity of analyzing possible states.

In this section, we introduce the building of individual trees, an intermediate tree, and find the possible states of ML-LTM. We also consider the importance of observed variables to reduce computation complexity. Since LTMs represent the similarities between observed variables, we can adopt the proposed ML-LTM in recommendation systems in Section IV.

IV. Multilayer Latent Tree Model for Recommendation Systems

With the explosive growth of information available on the Internet, recommendation systems have emerged from analyzing users' behavior and helping users to select relevant information. The correspondence of latent features between items and users can be discovered by LTM. Fig. 4 is an example of adopting an ML-LTM to make the recommendation. The product description is the input of the recommendation system. For example, the input data of the recommendation system are the descriptions of clothes. The ML-LTM is trained based on the observed variables, and the latent states are extracted from input descriptions. If a recommendation request, such as "Twill = Yes" is given, we compute the maximum JP of possible states by the ML-LTM. The relevant variables with different content such as "Blue" will be recommended to users. The detail of the recommendation process by ML-LTM is shown in Fig. 5. There are six steps in the recommendation process. To explain further, we define some parameters in Table IV.

A. Build ML-LTM

Based on the input data, the ML-LTM is built. The individual LTMs of different contents are built by the BI algorithm [12]. Intermediate latent tree connects the relationship of different contents.

TABLE IV
Definition of Parameters of Recommendation System

Parameter	Definition
SLV	The source latent variable of recommendation request.
C	The number of candidate latent variables.
CLV_c	The candidate latent variable, which is related to SLV.
S_{CLV}	The set of the candidate latent variables (CLV).
TLV	Target latent variable.
SS_{TLV}	The set of latent states of the target latent variable (TLV).
v_{NULL}	The state of the latent variable that all observed variables equal to *Null*.
N_{SLV}	The number of latent states of SLV.
N_{CLV_c}	The number of latent states of CLV_c.
$N_{observed}$	The number of observed variables.

TABLE V
Example of Meaningless Data

Data	O_1^1	...	O_5^1	O_1^2	...	O_4^2
$Data_1$	Yes	...	Yes	Yes	...	$NULL$
$Data_2$	$NULL$	...	$NULL$	$NULL$	...	$NULL$
...						

B. Find Meaningless States

Observed variables have various states, including "Null." "Null" represents that the observed variable does not exist in the data. When all observed variables are equal to *Null*, the state of the latent variable is called the meaningless state (v_{Null}). Table V is an example of meaningless data. All observed variables in Data_2 is "Null". The latent state of Data_2 is v_{Null}. In recommendation systems, we cannot recommend variables, which are equal to "Null," to users. We should remove v_{Null} before making a recommendation. The same as the transforming process in Section III-B, we calculate posterior probability of latent states to find v_{Null}. The maximum posterior probability among latent states is chosen to represent v_{Null}.

Assume observed variables $O_1^n, O_2^n, \ldots, O_i^n$ belong to L_k^n. u_i^n is the state of O_i^n. L_k^n has M_k^n latent states. $v_{k,1}^n, v_{k,2}^n, \ldots, v_{k,m}^n$ are latent states of L_k^n. The posterior probability of ($O_1^n = \text{Null}, O_2^n = \text{Null}, \ldots, O_i^n = \text{Null}$) with $L_k^n = v_{k,1}^n, L_k^n = v_{k,2}^n, \ldots, L_k^n = v_{k,m}^n$ are $P(x_1), P(x_2), \ldots, P(x_m)$, respectively. $P(x_1)$ can be represented as (12)

$$\begin{aligned} P(x_1) &= P\big(L_1^n = v_{1,1}^n \big| O_1^n = \text{Null}, \ \ldots, O_i^n = \text{Null}\big) \\ &= \frac{P\big(L_1^n = v_{1,1}^n, \ O_1^n = \text{Null}, \ \ldots, O_i^n = \text{Null}\big)}{P\big(O_1^n = \text{Null}, \ \ldots, O_i^n = \text{Null}\big)}. \end{aligned} \tag{12}$$

Since the denominator of $P(x_1), \ldots, P(x_m)$ is the same, we can simplify $P(x_1), \ldots, P(x_m)$ to (13). The maximum posterior probability of the latent states represents v_{Null}.

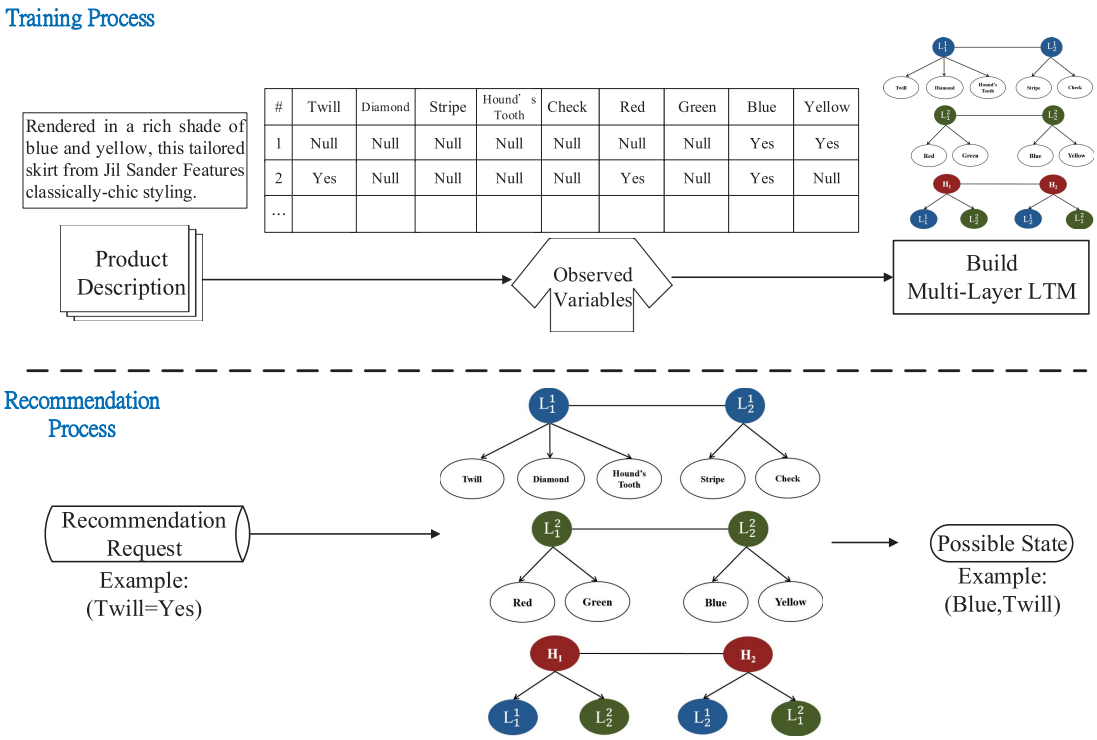

Fig. 4. Example of latent tree for recommendation system.

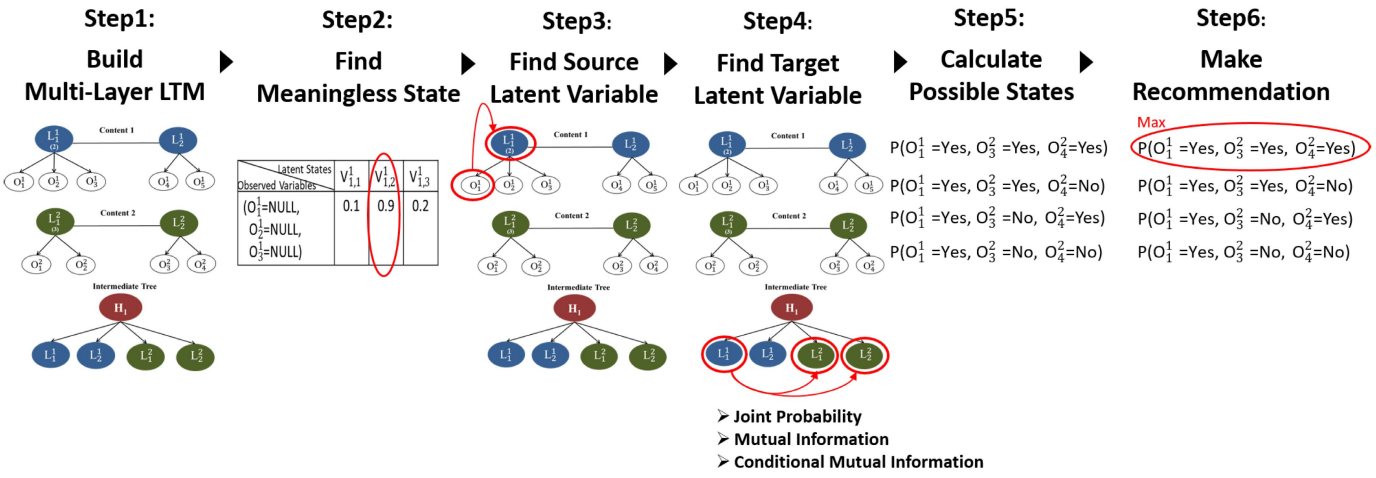

Fig. 5. Recommendation process of ML-LTM.

For example, if $P(x_1)$ has maximum posterior probability, $v^n_{1,1}$ represents v_{Null}. In the following "Find Target Latent Variable" process, we delete v_{Null} before the calculation. Through this process, we can only carry over useful states. The time complexity of this step is related to the number of latent states of L^n_k. We compare the posterior probability of all the latent states of L^n_k, as shown in (13). Therefore, assuming the number of the latent states of L^n_k is N, the time complexity is $O(N)$

$$\begin{aligned} P(x_1) &= P\left(L^n_1 = v^n_{1,1},\ O^n_1 = \text{Null},\ \ldots, O^n_i = \text{Null}\right) \\ &\cdots \\ P(x_m) &= P\left(L^n_1 = v^n_{1,m},\ O^n_1 = \text{Null},\ \ldots, O^n_i = \text{Null}\right). \end{aligned} \quad (13)$$

C. Find Source Latent Variable

The recommendation request is given in the third step. The parent latent variable of recommendation request is regarded as a source latent variable (SLV). In Section IV-D, we use Fig. 2 as the example. $O^1_1 = u^1_{1,1}$ is given as the recommendation request and SLV is L^1_1.

D. Find Target Latent Variable

The target latent variables (TLVs) are related to the recommendation request based on the ML-LTM and are utilized to do the recommendation. The ideal case of ML-LTM is shown in Fig. 2(c). We can find the relationship among

different contents via the intermediate tree. However, the difference between different contents in real applications is not big. Latent variables of different contents in intermediate tree grouped and belonged to one latent variable, as shown in Fig. 2(d). In this case, it is hard to find meaningful candidate latent variables (CLV_c), which are related to SLV. The computation process becomes complex. Therefore, JP, MI, and CMI are proposed in this step to find the TLV.

Follow the example of Fig. 2, L_1^1 is SLV. L_1^1 has two latent states $v_{1,1}^1$ and $v_{1,2}^1$. L_1^2 and L_2^2 are two CLV_c which are related to L_1^1. L_1^2 has two observed variables O_1^2 and O_2^2 and three latent states $v_{1,1}^2$,$v_{1,2}^2$, and $v_{1,3}^2$.

1) Joint Probability: The target function of finding TLV from S_{CLV_c} by JP is

$$F(\mathrm{SLV}, \mathrm{CLV}_c) = \arg\max_{1\le c\le C} P(\mathrm{SLV}, \mathrm{CLV}_c). \tag{14}$$

$O_1^1 = u_{1,1}$ is given as the recommendation request. The joint probabilities of SLV and two candidate latent variables L_1^2 and L_2^2 are calculated. The latent variable with the maximum JP will be chosen as TLV.

However, the joint probabilities of SLV and different CLV_c with latent states are equal to one since latent states of latent variables represent whole data. For example, in Fig. 2, $\sum_{v_{1,x}^1\in SS_{L_1^1},\, v_{1,y}^2\in SS_{L_1^2}} P(L_1^1 = v_{1,x}^1, L_1^2 = v_{1,y}^2)$ and $\sum_{v_{1,x}^1\in SS_{L_1^1},\, v_{2,y}^2\in SS_{L_2^2}} P(L_1^1 = v_{1,x}^1, L_2^2 = v_{2,y}^2)$ are all equal to one, where $SS_{L_1^1}$, $SS_{L_1^2}$, and $SS_{L_2^2}$ are the sets of latent states of L_1^1, L_1^2, and L_2^2, respectively. To make each JP different, we remove the meaningless state (v_{Null}) of each latent variable.

(13) is adopted to find the maximum posterior probability. We assume the posterior probability of ($O_1^2 = \mathrm{Null}$ and $O_2^2 = \mathrm{Null}$) with $L_1^2 = v_{1,1}^2, L_1^2 = v_{1,2}^2$, and $L_1^2 = v_{1,3}^2$ are $P(x_1)$, $P(x_2)$, and $P(x_3)$, respectively. $P(x_1)$, $P(x_2)$, and $P(x_3)$ are shown in (15).

$$\begin{aligned} P(x_1) &= P\big(L_1^2 = v_{1,1}^2\, O_1^2 = \mathrm{Null}\, O_2^2 = \mathrm{Null}\big) \\ P(x_2) &= P\big(L_1^2 = v_{1,2}^2,\ O_1^2 = \mathrm{Null},\ O_2^2 = \mathrm{Null}\big) \\ P(x_3) &= P\big(L_1^2 = v_{1,3}^2,\ O_1^2 = \mathrm{Null},\ O_2^2 = \mathrm{Null}\big). \end{aligned} \tag{15}$$

The maximum posterior probability is chosen as the latent state to represent v_{Null}. We assume that the maximum posterior probability is $P(x_1)$ and $v_{1,1}^2$ represents v_{Null}. After removing v_{Null}, we calculate JP with L_1^1 and the remaining states $v_{1,2}^2$ and $v_{1,3}^2$, as shown in (16)

$$\begin{aligned} P(X) &= \sum_{\substack{v_{1,x}^1\in SS_{L_1^1},\\ v_{1,y}^2\in SS_{L_1^2},\\ v_{1,y}^2\neq v_{\mathrm{NULL}}}} P\big(L_1^1 = v_{1,x}^1, L_1^2 = v_{1,y}^2\big) \\ &= p\big(L_1^1 = v_{1,1}^1, L_1^2 = v_{1,2}^2\big) + p\big(L_1^1 = v_{1,1}^1, L_1^2 = v_{1,3}^2\big) \\ &\quad + p\big(L_1^1 = v_{1,2}^1, L_1^2 = v_{1,2}^2\big) \\ &\quad + p\big(L_1^1 = v_{1,2}^1, L_1^2 = v_{1,3}^2\big). \end{aligned} \tag{16}$$

From (16), we can see that $P(X)$ is equal to 1-$P(x_1)$. In other words, finding maximum $P(X)$ is equivalent to finding the minimum $P(x_1)$, as follows:

$$\min \sum_{v_{1x}^1\in SS_{L_1^1}, v_{1,y}^2 = v_{\mathrm{NULL}}} p\big(L_1^1 = v_{1_x}^1, L_1^2 = v_{1,y}^2\big). \tag{17}$$

The number of computation process of (16) is $N_{\mathrm{SLV}} \times N_{\mathrm{CLV}_c}$, where N_{SLV} is the number of latent states of SLV and N_{CLV_c} is the number of latent states of CLV_c. Thus, the time complexity is $O(N * M)$, where N is the the number of latent states of SLV and M is the number of latent states of CLV_c. The number of computation process of (17) is $N_{\mathrm{SLV}} \times 1$ so the time complexity is $O(N)$. The number of computation process can be substantially decreased.

L_2^2 is another candidate latent variable. The JP of L_1^1 and L_2^2 is also calculated, as shown in (18). The time complexity is $O(N)$, where N is the number of latent states of L_1^1 and L_2^2

$$\min \sum_{v_{1,x}^1\in SS_{L_1^1}, v_{2,y}^2 = v_{\mathrm{NULL}}} p\big(L_1^1 = v_{1,x}^1, L_2^2 = v_{2,y}^2\big). \tag{18}$$

The JP of (17) and (18) are compared. The latent variable, which has the minimum JP, is chosen to be TLV. Therefore, the time complexity of finding TLV by JP is $O(N) + O(N) = O(N)$.

2) Mutual Information: MI in (3) can be adopted to determine TLV from CLV_i. The objective function is shown in (19)

$$\begin{aligned} &F(\mathrm{SLV}, \mathrm{CLV}_i) \\ &= \max_{\substack{v_x\in SS_{\mathrm{SLV}},\\ v_y\in SS_{\mathrm{CLV}_c},\\ 1\le c\le C}} I(\mathrm{SLV} = v_x, \mathrm{CLV}_c = v_y) \\ &= \max_{1\le c\le C} \sum_{v_x\in SS_{\mathrm{SLV}}, v_y\in SS_{\mathrm{CLV}_c}} P(\mathrm{SLV} = v_x, \mathrm{CLV}_c = v_y) \\ &\quad \times \log \frac{P(\mathrm{SLV} = v_x, \mathrm{CLV}_c = v_y)}{P(\mathrm{SLV} = v_x) P(\mathrm{CLV}_c = v_y)}. \end{aligned} \tag{19}$$

During the recommendation, if all observed variables are equal to *Null*, the recommendation accuracy becomes lower. Therefore, we find and remove v_{Null}.

The maximum posterior probability in (13) is chosen as the latent state to represent v_{Null}. If $P(x_1)$ has the maximum posterior probability, $v_{1,1}^2$ represents v_{Null}. Therefore, the MI of L_1^2 without v_{Null} is expressed in (20)

$$\begin{aligned} &I'\big(L_1^1 = v_{1,x}^1, L_1^2 = v_{1,y}^2\big) \\ &= \sum_{v_{1,x}^1\in SS_{L_1^1}, v_{1,y}^2\in SS_{L_1^2}, v_{1,y}^2\notin v_{1,1}^2} P\big(L_1^1 = v_{1,x}^1, L_1^2 = v_{1,y}^2\big) \\ &\quad \times \log \frac{P\big(L_1^1 = v_{1,x}^1, L_1^2 = v_{1,y^2}\big)}{P\big(L_1^1 = v_{1,x}^1\big) P\big(L_1^2 = v_{1,y}^2\big)}. \end{aligned} \tag{20}$$

The number of computation process is $N_{\mathrm{SLV}} \times (N_{\mathrm{CLV}_c} - 1)$. Thus, the time complexity is $O(N*M)$, where N is the number of latent states of SLV, and M is the number of latent states of CLV_c.

The final step is comparing MI of L_1^2 and L_2^2. The latent variable, which has the maximum MI, will be chosen as TLV. Therefore, the time complexity of finding TLV by MI is $O(N*M)+O(N*M)=O(N*M)$.

3) Conditional Mutual Information: Recommendation request is considered during the process of CMI. For example, when $O_1^1=u_{1,j}^1$ and $O_2^1=u_{2,j}^1$ are given as the recommendation request, the MI of O_1^1 and O_2^1 is the same. The CMI is different since O_1^1 and O_2^1 are considered in the probabilities. The CLV_c, which has the maximum CMI, will be defined as TLV. We assume the recommendation request be $O_i^n=u_{i,j}^n$ and the objective function of CMI is defined in (21)

$$\begin{aligned}
&F(\text{SLV},\text{CLV}_c)\\
&=\max_{\substack{v_x\in SS_{\text{SLV}},\\ v_y\in SS_{\text{CLV}_c},\\ 1\le c\le C}} I(\text{SLV}=v_x,\text{CLV}_c=v_y|\text{request})\\
&=\max_{1\le c\le C}\sum_{\substack{v_x\in SS_{\text{SLV}},\\ v_y\in SS_{\text{CLV}_c},}}\\
&\quad\times\left[P\left(\text{SLV}=v_x,\text{CLV}_c=v_y,O_i^n=u_{i,j}^n\right)\times\log\right.\\
&\qquad\left.\times\frac{P\left(O_i^n=u_{i,j}^n\right)P\left(\text{SLV}=v_x,\text{CLV}_c=v_y,O_i^n=u_{i,j}^n\right)}{P\left(\text{SLV}=v_x,O_i^n=u_{i,j}^n\right)P\left(\text{CLV}_c=v_y,O_i^n=u_{i,j}^n\right)}\right].
\end{aligned}\tag{21}$$

The same as the process of calculating MI, we consider CMI without v_{Null}. The process of finding v_{Null} is also the same as the process in Section IV-D.2. We assume $v_{1,1}^2$ represents v_{Null}. Therefore, the CMI of L_1^2 without v_{Null} is in (22)

$$\begin{aligned}
&I'\left(L_1^1=v_{1_x}^1,L_1^2=v_{2,1_y}|O_1^1=u_{1,j}^1\right)\\
&=\sum_{\substack{v_{1_x}^1\in SS_{L_1^1},\\ v_{2,1_y}\in SS_{L_1^2},\\ v_{2,1_y}\notin v_{1,1}^2}}\\
&\quad\times\left[P\left(L_1^1=v_{1_x}^1,L_1^2=v_{2,1_y},O_1^1=u_{1,j}^1\right)\times\log\right.\\
&\qquad\left.\times\frac{P\left(O_1^1=u_{1,j}^1\right)P\left(L_1^1=v_{1_x}^1,L_1^2=v_{2,1_y},O_1^1=u_{1,j}^1\right)}{P\left(L_1^1=v_{1_x}^1,O_1^1=u_{1,j}^1\right)P\left(L_1^2=v_{21_y},O_1^1=u_{1,j}^1\right)}\right].
\end{aligned}\tag{22}$$

The number of computation process is $N_{\text{SLV}}\times(N_{\text{CLV}_i}-1)$. Thus, the time complexity is $O(N*M)$, where N is the number of latent states of SLV, and M is the number of latent states of CLV_c.

The CMI of L_1^2 and L_2^2 is compared. The latent variable, which has maximum CMI is chosen as TLV. Therefore, the time complexity of finding TLV by CMI is $O(N*M)+O(N*M)=O(N*M)$.

E. Calculate Possible States

After finding TLV, the possible states are calculated by JP in.(9). The MI in (3) and IC in (11) are also adopted to differentiate the "importance" of observed variables and decrease the computation process. Assume L_2^2 is TLV. The possible states of $O_1^1=u_{1,j}^1$, O_3^2, and O_4^2 are calculated in (23)

$$\begin{aligned}
&P\left(O_1^1=u_{1,j}^1|L_2^2\right)\\
&=P\left(O_1^1=u_{1,j}^1|O_3^2,O_4^2\right)\\
&=P\left(O_1^1=u_{1,j}^1|L_2^2\right)\times P\left(L_2^2|H_1\right)\times P(H_1)\times P\left(L_2^2|H_1\right)\\
&\quad\times P\left(O_3^2|L_2^2\right)\times P\left(O_4^2|L_2^2\right)\\
&=P\left(O_1^1=u_{1,j}^1,L_2^2\right)\times P\left(O_3^2=u_{3,j}^2|L_2^2\right)\\
&\quad\times P\left(O_4^2=u_{4,j}^2|L_2^2\right).
\end{aligned}\tag{23}$$

F. Make Recommendation

The possible state $(O_3^2=u_{3,j}^2,O_4^2=u_{4,j}^2)$ with the maximum JP in (24) will be the recommendation result

$$\max P\left(O_1^1=u_{1,j}^1,L_2^2\right)P\left(O_3^2=u_{3,j}^2|L_2^2\right)P\left(O_4^2=u_{4,j}^2|L_2^2\right).\tag{24}$$

We demonstrate how to adopt ML-LTM in the recommendation system. By adopting JP, MI, and CMI, the TLV is found, and the latent states of TLV are utilized for a recommendation. The computation complexity is reduced, and the recommendation accuracy is increased by the proposed process.

V. Incremental Update Techniques for Multilayer Latent Tree

The original LTM built in the batch mode may become obsolete when new data are given as input. The accuracy of LTM will be intensively decreased. Retraining LTM for updating the system consumes much time. An efficient updating method for LTM is necessary.

Before designing the updating techniques, we observe that when new input data come in, the observed variable that has less correlation with the parent latent variable may jump to other latent variables. We define these observed variables as unstable variables. Therefore, we identify unstable variables and update these variables to speed up the model updating. Preparing stage and updating model stage are two stages for incrementally updating the latent tree.

A. Preparing Stage

In the preparation stage, unstable observed variables are found and labeled by IC and MI. We assume that O_1^1 is an observed variable, and O_1^1 belongs to latent variable L_1^1. IC in (11) can distinguish the importance of observed variables. The IC of the stable variable is large. If IC is larger than the threshold, it means that there is a great deal of information between O_1^1 and L_1^1. O_1^1 is important to L_1^1. MI represents the similarity between the observed variable and the latent variable. If the observed variable is not similar to the parent latent variable, the observed variable is unstable, as follows:

$$\begin{aligned}
&\text{I}\left(O_1^1,L_1^1\right)-\text{I}\left(O_1^1,L\right)>\text{MI}_{\text{threshold}}\\
&\qquad\forall L\in\text{The set of latent variable},\quad L\neq L_1^1.
\end{aligned}\tag{25}$$

If the IC of O_1^1 is not larger than threshold and MI of O_1^1 cannot satisfy (25), O_1^1 is tagged as an unstable variable.

B. Updating Model Stage

In this stage, there are three steps to check the status of unstable observed variables and update the model incrementally. The first step is updating the tree structure based on the new data. As mentioned before, the number of latent variables, the number of latent states, and the conditional probabilities between variables are three necessary information of an LTM. According to different information of original LTM is retained, there are three methods to update LTM: 1) LCM; 2) EM method; and 3) copy method.

When updating the latent tree, we utilize LCM to keep the number of latent variables. In Fig. 2, assume the number of latent states of L_1^1 becomes three. The conditional probabilities among observed variables and latent variables are recomputed in (26), where $u_{1,1}^1,\ u_{1,2}^1,\ u_{1,3}^1$ are new latent states of L_1^1

$$\begin{aligned}
&P\big(L_1^1 = u_{1,1}^1\big), \quad P\big(L_1^1 = u_{1,2}^1\big),\ P\big(L_1^1 = u_{1,3}^1\big)\\
&P\big(O_1^1 \big| L_1^1 = u_{1,1}^1\big), \quad P\big(O_1^1 \big| L_1^1 = u_{1,2}^1\big),\ P\big(O_1^1 \big| L_1^1 = u_{1,3}^1\big)\\
&P\big(O_2^1 \big| L_1^1 = u_{1,1}^1\big), \quad P\big(O_2^1 \big| L_1^1 = u_{1,2}^1\big),\ P\big(O_2^1 \big| L_1^1 = u_{1,3}^1\big)\\
&P\big(O_3^1 \big| L_1^1 = u_{1,1}^1\big), \quad P\big(O_3^1 \big| L_1^1 = u_{1,2}^1\big),\ P\big(O_3^1 \big| L_1^1 = u_{1,3}^1\big).
\end{aligned} \tag{26}$$

If we record the number of latent variables and the number of latent states to update the tree model, it is the EM method. In Fig. 2, the number of latent states of L_1^1 is still two. The conditional probability between observed and latent variables based on new data is recalculated, as shown in (27)

$$\begin{aligned}
&P\big(L_1^1 = u_{1,1}^1\big), \quad P\big(L_1^1 = u_{1,2}^1\big)\\
&P\big(O_1^1 \big| L_1^1 = u_{1,1}^1\big), \quad P\big(O_1^1 \big| L_1^1 = u_{1,2}^1\big)\\
&P\big(O_2^1 \big| L_1^1 = u_{1,1}^1\big), \quad P\big(O_2^1 \big| L_1^1 = u_{1,2}^1\big).
\end{aligned} \tag{27}$$

When the number of latent variables, the number of latent states, and the conditional probabilities between variables are retained to update the tree, it is the copy method.

The second step is computing MI among unstable variables and latent variables based on the new data. If the MI is higher than the current value, the unstable variable jumps to the other latent variable. In this step, we use MI to find the new latent variable for the unstable variable.

The third step connects the subtrees and optimizes new LTM by the Chow–Liu and EM algorithms.

VI. Experiments

In this section, we evaluate the performance of the proposed ML-LTM and the incremental update mechanism. We develop ML-LTM for the recommendation system and compare the recommendation accuracy with latent semantic analysis (LSA) and various popular matrix factorization methods. In addition, we compare the accuracy and computation time of the proposed incremental update method and updating the whole LTM method.

Jaccard similarity in (28) is adopted to compare the recommendation accuracy of two considered methods. The Jaccard similarity is larger if more recommended features are matched with the real data

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{28}$$

We also adopt precision in (29) and recall in (30) to further discuss the recommendation result [24]

$$\text{Precision} = \left| \frac{\text{Interesting} \cap \text{TopN}}{N} \right| \tag{29}$$

$$\text{Recall} = \left| \frac{\text{Interesting} \cap \text{TopN}}{|\text{Interesting}|} \right|. \tag{30}$$

In the evaluation of Jaccard similarity, precision, and recall, we only consider whether the recommended feature is matched with the real data. However, the ranking of recommendation items decides the quality of recommendation. Thus, we measure the discounted cumulative gain (DCG) to evaluate the "ranking" quality. DCG measures the usefulness of an item based on its position in the recommendation list [25]. In DCG, the recommended items which are more relevant to users are given higher gain. We define recommended items with higher order position in the recommended list to be more relevant items to users in the experiments. The DCG equation is shown in the following:

$$\text{DCG}_k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)} \tag{31}$$

where k represents the number of items that is recommended and rel_i is gain for each recommendation item. In the experiments, we use Jaccard similarity to be rel_i. The DCG of ideal recommendation order is defined as ideal DCG (IDCG). We divide the DCG value by IDCG, which is called normalized DCG (NDCG), to show the result.

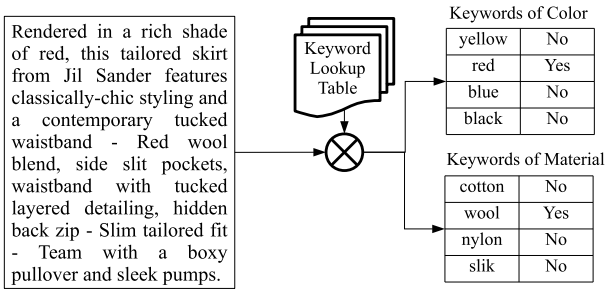

Fig. 6. Keyword mapping flow of Polyvore data.

A. Data Set Preparation

We use three data sets in the following experiments. The open data set is from [26], including "Asia, Alarm, Coil, News, and WebKb" data set. We collect the second data set from Polyvore [27]. Polyvore is a social platform for users to share favorite clothes. We collect the description of cloth data and map data with a keyword table to transform text data into classification data, as shown in Fig. 6. For example, if we want to analyze color data of clothes, we find "Red, Green, Yellow, and so on" by the color keyword table and record whether the description includes these keywords in the mapping result table.

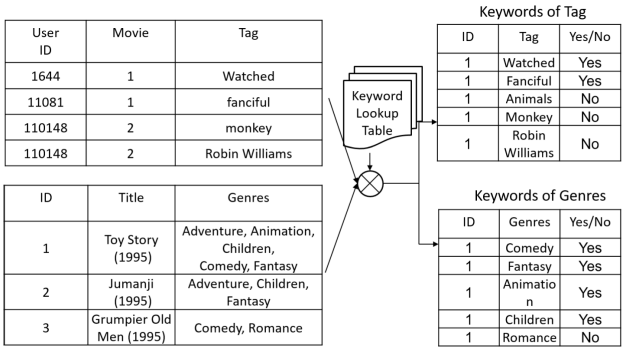

Fig. 7. Keyword mapping flow of Movielens data.

TABLE VI
INFORMATION OF DATA SET

Dataset	The Number of Observed Variables	The Number of Data	Data Type
Asia	7	200	Binary Data
Alarm	37	2000	Category Data
Coil	42	9822	Category Data
News	100	16242	Binary Data
WebKb	300	1500	Binary Data
Polyvore-Color	8	Larger than 100MB	Category Data
Polyvore-Material	47	Larger than 100MB	Binary Data
Polyvore-Pattern	155	Larger than 100MB	Binary Data
Movielens	465,000	20 Million	Binary Data

Besides Polyvore, we also utilize Movielens [28], which is the most popular data set to evaluate recommendation accuracy. We adopt "MovieLens 20M Data Set," which contains 20 million ratings and 465 000 tag applications applied to 27 000 movies by 138 000 users. To transform the descriptions and the tags of movies into binary data, we analyze the top 100 most frequently occurring words and record whether the descriptions include these words, as shown in Fig. 7.

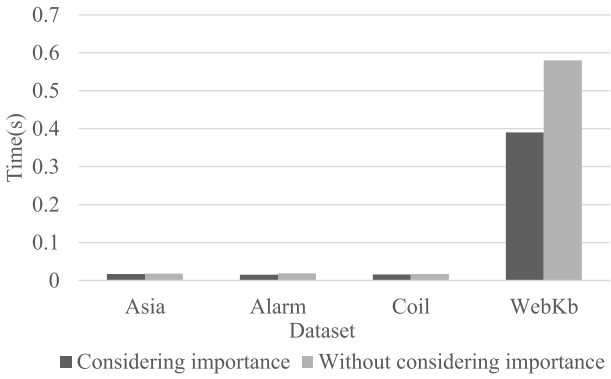

Fig. 8. Computation time of considering importance variables.

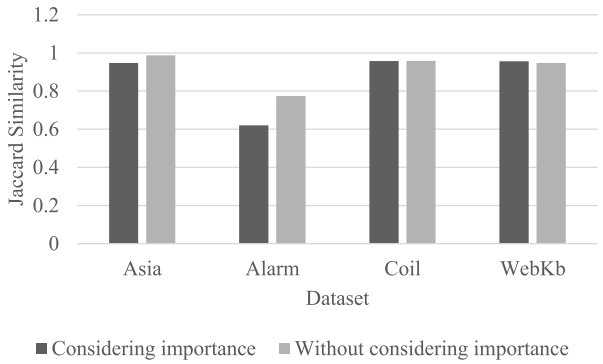

Fig. 9. Accuracy of considering importance of variables.

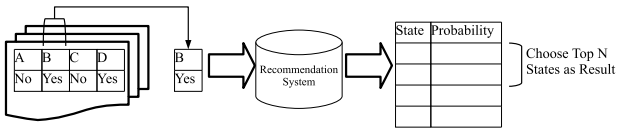

Fig. 10. Flow of single-content recommendation.

Table VI is a summary of the data set. The data are divided into 50% training data and 50% test data. The training data train the initial model. The new streaming data are composed of the different proportion of training data and testing data to evaluate the accuracy of the updating model.

B. Importance of Observed Variables

In the proposed ML-LTM, we consider the "importance" of an observed variable to decrease the computation complexity of finding latent states. Figs. 8 and 9 show that considering the importance of observed variables can intensively decrease the computation time but is not affected the recommendation accuracy, especially in the WebKb data set, which contains the largest number of observed variables. Therefore, we will consider the importance of observed variables in the following experiments.

C. Multilayer LTM for Recommendation Systems

To demonstrate the proposed LTM can work in a recommendation system, we compare the proposed LTM recommendation system with LSA.

1) Single-Content Recommendation: LSA decomposes document files into the matrix of latent factors by singular-value

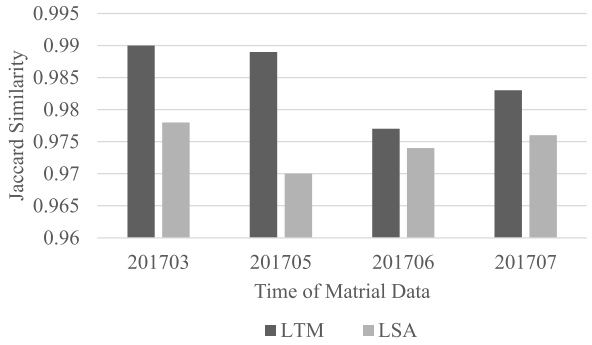

Fig. 11. Jaccard similarity of LSA and LTM over the material data.

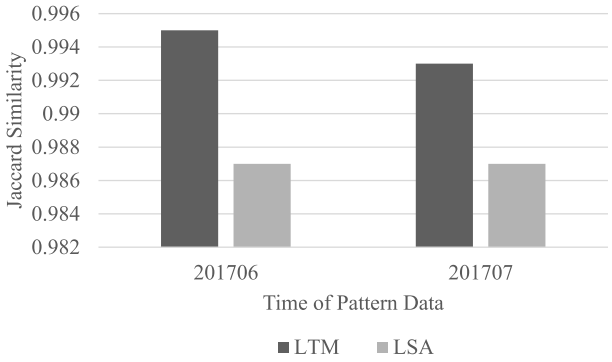

Fig. 12. Jaccard similarity of LSA and LTM over the pattern data.

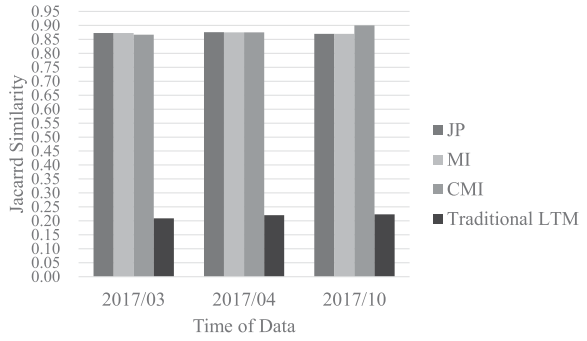

Fig. 13. Jaccard similarity of JP, MI, and CMI over the Polyvore data set.

decomposition to recommend similar documents based on latent factors. Since LSA can only recommend single content, we compare LTM with LSA in single-content recommendation based on Polyvore data, as shown in Fig. 10. A feature is randomly selected as the recommendation request. LTM finds the sibling observed variables of recommendation request and calculates the JP. According to the value of JP, the top-n combination of observed variables will be recommended. N is set to five in the following experiments.

Figs. 11 and 12 show the Jaccard similarities of recommending material data and pattern data by LTM and LSA, respectively. The recommendation result of LTM is 98% similar to real data. Compared with LSA, LTM derives more accurate recommendation results.

2) Multicontent Recommendation: We want to compare the recommendation accuracy between the proposed ML-LTM and the traditional LTM when we make the multicontent recommendation. The traditional LTM can only deal with single-content data. Thus, the recommendation items are randomly chosen in the traditional LTM. The recommendation items of ML-LTM are chosen by the JP, MI, and CMI, which are introduced in Section IV. We evaluate the recommendation performance of JP, MI, and CMI over Polyvore and Movielens data sets, respectively. Fig. 13 shows the Jaccard similarity of JP, MI, and CMI in ML-LTM over Polyvore data. The Jaccard similarity of proposed JP, MI, and CMI can achieve about 90% accuracy and is much higher than existing LTM.

TABLE VII
COMPARISON OF JP, MI, AND CMI OVER MOVIELENS DATA SET

Algorithm	Jaccard Similarity	Precision	Recall
JP	0.87	0.83	0.36
MI	0.9	0.88	0.68
CMI	**0.9**	**0.96**	**0.76**
Traditional LTM	0.74	0.33	0.20

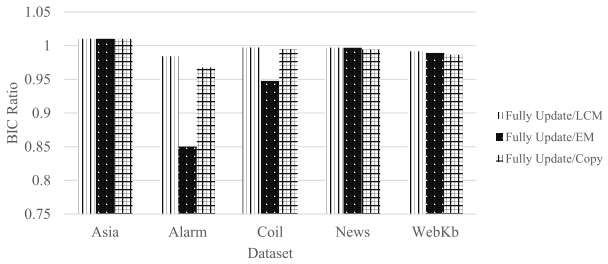

Fig. 14. BIC ratio of updating methods over different open data sets.

Based on Movielens data sets, Table VII shows that MI and CMI can get higher Jaccard similarity, precision, and recall. The reason is that MI and CMI consider the latent topics when calculating the latent states.

To further evaluate the recommendation performance of JP, MI, and CMI, we compare these methods to different matrix factorization methods (PopRank, WR-MF, BPR-MF, ShiftMC, and CSRR) over Movielens dataset. In the recommendation list, the ranking of items is important. Thus, NDCG is utilized in this part to represent the recommended quality. NDCG@K shows a different number of recommendation items. PopRank is regarded as a baseline since PopRank recommends items to users based on the popularity of the items. Weighted regularized matrix factorization (WR-MF) [29] is a powerful matrix factorization model for item prediction on implicit feedback datasets. Bayesian personalized ranking matrix factorization (BPR-MF) [30] considers the ranking with implicit feedback when recommending items to users. Shift matrix completion (ShiftMC) [31] designed positive-unlabeled learning algorithm for matrix completion. [32] proposed a robust cost sensitive learning for recommendation (CSRR) with implicit feedback. CSRR issued an update when an error occurs. We give movie tags as the recommendation request and recommend movie types to users. In Table VIII, the NDCG of JP, MI, and CMI is higher than 0.9 and much higher than other methods. That is, JP, MI, and CMI can recommend items which really relevant to users.

TABLE VIII
NDCG EVALUATION OVER MOVIELENS DATA SET

Algorithm	NDCG@5	NDCG@10	NDCG@15
JP	0.91	0.96	0.96
MI	0.91	0.93	0.94
CMI	**0.91**	**0.94**	**0.94**
Traditional LTM	0.59	0.66	0.73
PopRank	0.42	0.38	0.43
WR-MF	0.64	0.66	0.65
BPR-MF	0.61	0.62	0.65
ShiftMC	0.63	0.65	0.62
CSRR	0.62	0.65	0.7

D. Updating Experiment

We compare the accuracy and computation time of the proposed incremental update method and updating the whole LTM method. We call updating the whole LTM as "the fully update method" in the following experiments. We utilize the ratio of BIC score in (32) and Jaccard similarity in (28) to compare different updating methods. Since the BIC score is negative, $\text{BIC}_{\text{ratio}} > 1$ represents that the accuracy of incremental update mechanism is better

$$\text{BIC}_{\text{ratio}} = \frac{\text{BIC}_{\text{Fully Update}}}{\text{BIC}_{\text{Incremental Update}}}. \tag{32}$$

1) Preliminary Experiment: LCM, EM, and Copy are three methods in the proposed incremental update mechanism for LTM. In the preliminary experiment, we compare the accuracy and computation time of LCM, EM, copy methods, and the full update method based on open data. In Fig. 14, the BIC ratio of LCM, EM, and copy is close to 1. That is, the proposed LCM, EM, and copy of the incremental update mechanism can achieve equivalent accuracy to that of the fully update method.

In Fig. 15, the computation time of the copy method is only 4% of the computation time of the fully update method based on the Asia data set. In other data sets, the computation time of the fully update method is five times longer than the computation time of the LCM, EM, and copy methods. All in all, the proposed incremental update methods not only achieve comparable accuracy with the fully update method but also spend 20% computation time of the fully update method.

Since the accuracy of the LCM method is most similar to that of the fully update method and the computation time is not much different from other methods. Therefore, we choose the LCM method to compare accuracy and computation time with the fully update method in the following updating experiments.

2) Accuracy: We evaluate the recommendation accuracy of different updating methods based on the Polyvore data set. We use spring data and winter data, which are collected in April and November in 2016, to be the training data. Data of the following months are simulated as new input data to

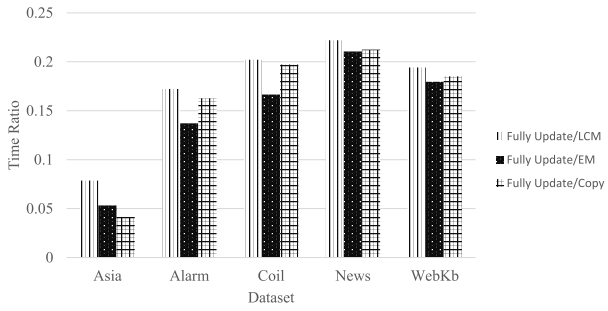

Fig. 15. Time ratio of updating methods over different open data sets.

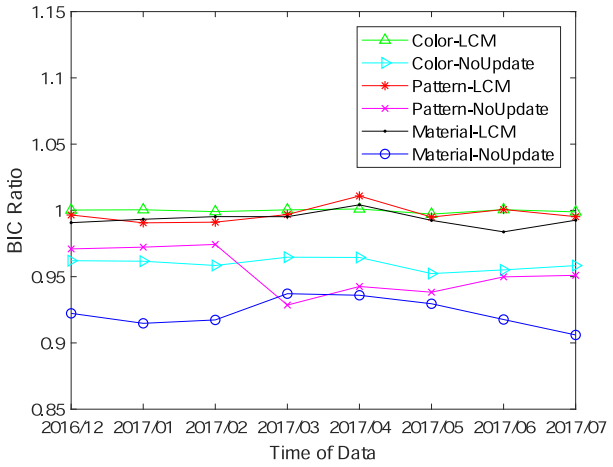

Fig. 16. BIC ratio of training ML-LTM by spring data.

LTM. Figs. 16 and 17 show the comparison of the BIC ratio. With training LTM based on spring data, the proposed LCM updating method makes LTM be similar to the LTM, which is fully updated, as shown in Fig. 16. If LTM is not updated, the accuracy of color, material, and pattern data is decreasing in winter and summer. The lowest accuracy of color data is in November. The lowest accuracy of material data is in January. When spring comes next year, the accuracy becomes higher. We also train the LTM model based on winter data, as shown in Fig. 17. We can see that the accuracy of LTM without updating decreases in spring and summer. The proposed LCM makes the accuracy of LTM keep close to one.

Figs. 18 and 19 show the comparison of the Jaccard similarity. With training LTM based on spring data, the Jaccard similarity of material data is more stable since the number of material variables is less, and the variance among variables is little. On the other hand, based on color data, if LTM is not updated, the Jaccard similarity is decreasing. With training LTM based on winter data, the Jaccard similarity of the proposed LCM is higher and more stable than LTM, which is fully updated and without updating.

3) Time: The ratio of computation time (33) is adopted to compare the computation time of incremental update methods and the fully update method

$$\text{Time}_{\text{ratio}} = \frac{\text{Time}_{\text{Incremental Update}}}{\text{Time}_{\text{Fully Update}}}. \tag{33}$$

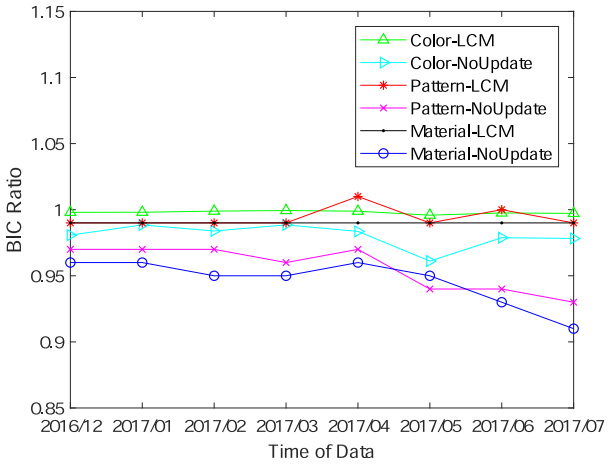

Fig. 17. BIC ratio of training LTM by winter data.

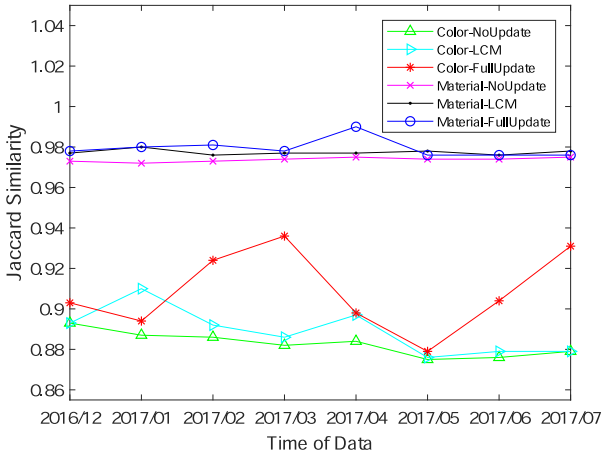

Fig. 18. Jaccard similarity of training LTM by spring data.

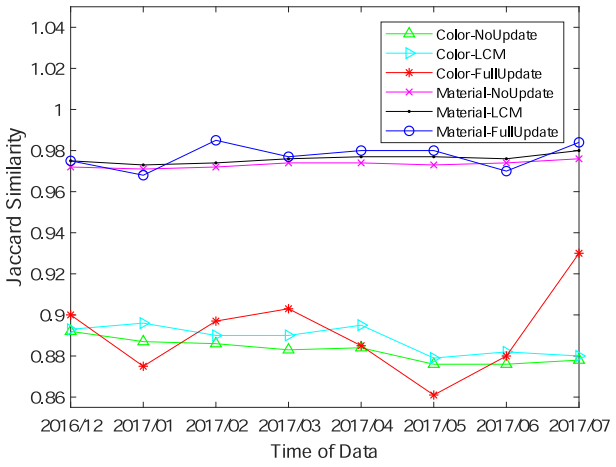

Fig. 19. Jaccard similarity of training LTM by winter data.

If the value of (33) is smaller than 1, the computation time of incremental update mechanism is less. Otherwise, the fully update method spends less time.

Figs. 20 and 21 show the time comparison on Polyvore data. We find that LCM spends 17% updating time of the fully update method. Especially in terms of material data, LCM saves much updating time. Since the materials of the clothes

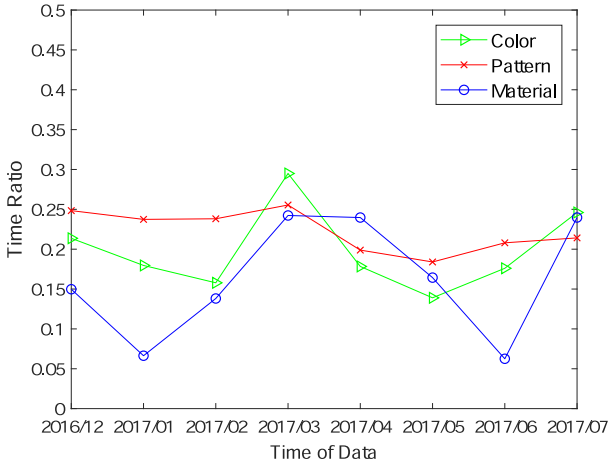

Fig. 20. Time ratio of training LTM by spring data.

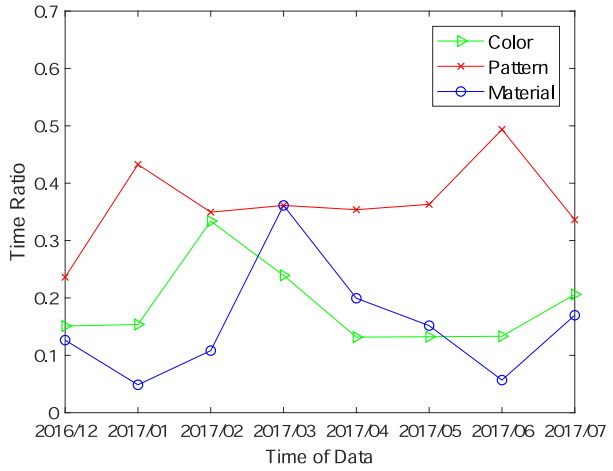

Fig. 21. Time ratio of training LTM by winter data.

we wear in every season are very different, the accuracy of material data fluctuates as the season changes. LTM of material data should update in every month. In this case, the proposed incremental update mechanism for LTM can intensively decrease the updating time and maintain accuracy.

4) Discussion: In the recommendation experiments, we design the single-content recommendation and the multicontent recommendation evaluation. In the single-content recommendation, we compare the proposed ML-LTM to LSA. LTM can achieve 98% similar to real data.

In the multicontent recommendation, we evaluate the performance of three proposed recommendation methods (JP, MI, and CMI) and the traditional LTM. The Jaccard similarity of the proposed JP, MI, and CMI can achieve about 90% accuracy and is much higher than the existing LTM. Besides, MI and CMI can achieve more accurate results since MI and CMI consider the latent topics when during the calculation of latent states. We also compare JP, MI, and CMI to various popular matrix factorization methods (PopRank, WM-RF, BPR-MF, ShiftMC, and CSRR). The NDCG of JP, MI, and CMI is higher than 0.9 and much higher than other methods. In ML-LTM, we have grouped the related variables. Therefore, ML-LTM can find recommendation variables that are more

relevant to the recommendation request. All in all, ML-LTM can be effectively adopted in the recommendation system to recommend the related items.

In the updating experiment, we compare the accuracy and computation time of the proposed incremental update method and updating the whole LTM method. In the preliminary experiments, we find that among the proposed updating approaches (LCM, EM, and copy methods), the accuracy of the LCM method is most similar to that of the fully update method and the computation time is not much different from other methods. Since LCM only retain the number of latent variables, LCM can more easily adapt to the new data and obtain higher accuracy. Besides, in the updating experiments, the proposed LCM updating method makes the accuracy of LTM keep close to one and only spend 17% updating time of the fully update method. Since we only retain the number of latent variables and only update unstable variables, LCM can achieve effective model updating and retain accuracy.

VII. Conclusion

In this article, we investigated the modeling and updating issue of the latent tree for multicontent data. We first developed an ML-LTM to represent the relationship of different contents. Second, we further designed the incremental update mechanism to accelerate the updating process of ML-LTM. In the experiment of a single-content recommendation system, ML-LTM achieved 98% recommendation accuracy, which was much better than the accuracy of LSA. In the case of the multicontent recommendation system, ML-LTM had 90% accuracy, while the existing LTM could only achieve 20% accuracy. Besides, the NDCG of ML-LTM is higher than 0.9 and much higher than the current matrix factorization methods.

To address the model updating issue, we showed that the proposed incremental update mechanism spent only 17% update time of the fully update method under the condition of maintaining a similar accuracy. In the future, it is worthwhile extending the incremental updating approach of the ML-LTM to various latent factor models.

References

[1] Z. Han, M. Hong, and D. Wang, *Signal Processing Network for Big Data Application*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[2] D. Doran, S. Schulz, and T. R. Besold, "What does explainable AI really mean? A new conceptualization of perspectives," 2017, *arXiv:1710.00794*. [Online]. Available: http://arxiv.org/abs/1710.00794

[3] Y. Tao, Y. Jia, N. Wang, and H. Wang, "The fact: Taming latent factor models for explainability with factorization trees," in *ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2019, pp. 1–8.

[4] C.-Y. Lin, L.-C. Wang, and K.-H. Tsai, "Hybrid real-time matrix factorization for implicit feedback recommendation systems," *IEEE Access*, vol. 6, pp. 21369–21380, 2018.

[5] J. Han, L. Zheng, H. Huang, Y. Xu, P. S. Yu, and W. Zuo, "Deep latent factor model with hierarchical similarity measure for recommender systems," *Inf. Sci.*, vol. 503, pp. 521–532, Nov. 2019.

[6] N. L. Zhang, "Hierarchical latent class models for cluster analysis," *J. Mach. Learn. Res.*, vol. 5, pp. 697–723, Dec. 2004.

[7] R. Mourad, C. Sinoquet, N. L. Zhang, T. Liu, and P. Leray, "A survey on latent tree models and applications," *J. Artif. Intell. Res.*, vol. 47, pp. 157–203, May 2013.

[8] N. L. Zhang and L. K. Poon, "Latent tree analysis," in *Proc. AAAI*, 2017, pp. 1–5.

[9] L. M. Collins and S. T. Lanza, *Latent class latent transition analysis: With Appl. social, Behav., health Sci.*, vol. 718. Hoboken, NJ, USA: Wiley, 2013.

[10] N. L. Zhang, S. Yuan, T. Chen, and Y. Wang, "Latent tree models and diagnosis in traditional chinese medicine," *Artif. Intell. Med.*, vol. 42, no. 3, pp. 229–245, Mar. 2008.

[11] Y. Zhao, N. L. Zhang, T. Wang, and Q. Wang, "Discovering symptom co-occurrence patterns from 604 cases of depressive patient data using latent tree models," *J. Alternative Complementary Med.*, vol. 20, no. 4, pp. 265–271, Apr. 2014.

[12] T.-F. Liu, N. L. Zhang, P. Chen, A. H. Liu, L. K. M. Poon, and Y. Wang, "Greedy learning of latent tree models for multidimensional clustering," *Mach. Learn.*, vol. 98, nos. 1–2, pp. 301–330, Jan. 2015.

[13] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky, "Consistent and efficient reconstruction of latent tree models," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2010, pp. 719–725.

[14] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, Mar. 1978.

[15] N. L. Zhang and T. Kocka, "Effective dimensions of hierarchical latent class models," *J. Artif. Intell. Res.*, vol. 21, pp. 1–17, Jan. 2004.

[16] T. Chen, N. L. Zhang, T. Liu, K. M. Poon, and Y. Wang, "Model-based multidimensional clustering of categorical data," *Artif. Intell.*, vol. 176, no. 1, pp. 2246–2269, Jan. 2012.

[17] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 462–467, May 1968.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. society. Ser. B Methodol.*, vol. 39, pp. 1–38, Sep. 1977.

[19] T. M. Cover and J. A. Thomas, *Elements Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[20] X. Yang *et al.*, "Identifying the zheng in psoriatic patients based on latent class analysis of traditional chinese medicine symptoms and signs," *Chin. Med.*, vol. 9, no. 1, p. 1, 2014.

[21] T. Liu, N. L. Zhang, and P. Chen, "Hierarchical latent tree analysis for topic detection," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2014, pp. 256–272.

[22] Z. Liu, H. Chen, J. Li, and Y. Yu, "PWA-PEM for latent tree model and hierarchical topic detection," in *Proc. Int. Conf. Intell. Inf. Process.*, 2018, pp. 183–191.

[23] C.-Y. Lin, L.-C. Wang, and S.-P. Chang, "Incremental checkpointing for fault-tolerant stream processing systems: A data structure approach," *IEEE Trans. Emerg. Topics Comput.*, early access, Apr. 22, 2020, doi: 10.1109/TETC.2020.2986487.

[24] H. Zhu *et al.*, "Learning tree-based deep model for recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1079–1088.

[25] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2012, pp. 143–152.

[26] *Open Data*. Accessed: Jun. 2013. [Online]. Available: http://www.cse.ust.hk/~lzhang/ltm/softwares/BI.zip

[27] *Polyvore*. Accessed: Feb. 2007. [Online]. Available: https://www.polyvore.com/

[28] *Movielens Dataset*. Accessed: Apr. 2015. [Online]. Available: https://grouplens.org/datasets/movielens/20m/

[29] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 263–272.

[30] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*. [Online]. Available: http://arxiv.org/abs/1205.2618

[31] C.-J. Hsieh, N. Natarajan, and I. Dhillon, "Pu learning for matrix completion," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2445–2453.

[32] P. Yang, P. Zhao, Y. Liu, and X. Gao, "Robust cost-sensitive learning for recommendation with implicit feedback," in *Proc. SIAM Int. Conf. Data Mining*, 2018, pp. 621–629.

Chia-Yu Lin received the B.S. and M.S. degrees in computer science from National Chiao Tung University Guangfu Campus (NCTU), Hsinchu, Taiwan, R. O. C., in 2010 and 2012, respectively, and the Ph.D. degree from the Institute of Communications Engineering, NCTU, in 2019. She is currently a Researcher with NCTU.

Her current research interests include real-time updating techniques for recommendation algorithms and mathematical framework for data streaming applications.

Yu-Fang Chiu received the B.S. and M.S. degrees in electrical and computer engineering from National Chiao Tung University Guangfu Campus (NCTU), Hsinchu, Taiwan, in 2014 and 2018, respectively.

She is currently an Assistant Researcher with NCTU. Her current research interests include interference management for drone network communications.

Li-Chun Wang (Fellow, IEEE) received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, R. O. C., in 1986, the M.S. degree from National Taiwan University, Taipei, Taiwan, in 1988, and the M.Sc. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, GA, USA, in 1995 and 1996, respectively, all in electrical engineering.

From 1990 to 1992, he was with the Telecommunications Laboratories, Ministry of Transportations and Communications (currently, Chunghwa Telecom Laboratories of Chunghwa Telecom Company, Ltd.), Taoyuan City, Taiwan. In 1995, he was with Bell-Northern Research of Northern Telecom, Inc., Richardson, TX, USA. From 1996 to 2000, he was a Senior Technical Staff Member with the Wireless Communications Research Department, AT&T Laboratories, Florham Park, NJ, USA. Since 2000, he has been with National Chiao Tung University, Hsinchu, Taiwan, where he was the Chair of the Department of Electrical and Computer Engineering (ECE), where he is jointly appointed by the Department of ECE and the Department of Computer Science. He holds 18 U.S. patents. He has authored or coauthored over 90 journal articles, 180 conference articles, and co-edited a book, *Key Technologies for 5G Wireless Systems* (Cambridge, 2016). His recent research interests are focused on the cross-layer optimization and big-data-driven methodology for mobile broadband networks, and cognitive communications/computing for latency-critical Internet of Things applications.

Dr. Wang was elected to the IEEE Fellow for his contributions to cellular architectures and radio resource management in wireless networks. He received the 1997 IEEE Jack Neubauer Best Paper Award in 1997, the Distinguished Research Award of the National Science Council, Taiwan, in 2012, and the IEEE Communications Society Asia–Pacific Board Best Paper Award in 2015.

Dusit Niyato (Fellow, IEEE) received the B.Eng. degree from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 1999, and the Ph.D. degree in electrical and computer engineering from the University of Manitoba, Winnipeg, MB, Canada, in 2008.

He is currently a Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests are in the area of energy harvesting for wireless communication, Internet of Things, and sensor networks.