Context-aware Tree-based Deep Model for Recommender Systems

Daqing Chang*
Alibaba Group
daqing.cdq@alibaba-inc.com

Jintao Liu*
Tsinghua University
liujt17@mails.tsinghua.edu.cn

Ziru Xu*
Alibaba Group
ziru.xzr@alibaba-inc.com

Han Li
Alibaba Group
lihan.lh@alibaba-inc.com

Han Zhu
Alibaba Group
zhuhan.zh@alibaba-inc.com

Xiaoqiang Zhu
Alibaba Group
xiaoqiang.zxq@alibaba-inc.com

ABSTRACT

How to predict precise user preference and how to make efficient retrieval from a big corpus are two major challenges of large-scale industrial recommender systems. In tree-based methods, a tree structure $\mathcal{T}$ is adopted as index and each item in corpus is attached to a leaf node on $\mathcal{T}$. Then the recommendation problem is converted into a hierarchical retrieval problem solved by a beam search process efficiently.

In this paper, we argue that the tree index used to support efficient retrieval in tree-based methods also has rich hierarchical information about the corpus. Furthermore, we propose a novel context-aware tree-based deep model (ConTDM) for recommender systems. In ConTDM, a context-aware user preference prediction model $\mathcal{M}$ is designed to utilize both horizontal and vertical contexts on $\mathcal{T}$. Horizontally, a graph convolutional layer is used to enrich the representation of both users and nodes on $\mathcal{T}$ with their neighbours. Vertically, a parent fusion layer is designed in $\mathcal{M}$ to transmit the user preference representation in higher levels of $\mathcal{T}$ to the current level, grasping the essence that tree-based methods are generating the candidate set from coarse to detail during the beam search retrieval. Besides, we argue that the proposed user preference model in ConTDM can be conveniently extended to other tree-based methods for recommender systems. Both experiments on large scale real-world datasets and online A/B test in large scale industrial applications show the significant improvements brought by ConTDM.

CCS CONCEPTS

• **Computing methodologies → Classification and regression trees**; **Neural networks**; • **Information systems → Recommender systems**.

KEYWORDS

recommender systems, context-aware model, tree-based retrieval, large-scale problem

ACM Reference Format:
Daqing Chang*, Jintao Liu*, Ziru Xu, Han Li, Han Zhu, and Xiaoqiang Zhu. 2021. Context-aware Tree-based Deep Model for Recommender Systems. In *Proceedings of DLP-KDD 2021.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommendation problem is generally to retrieve for a candidate set comprised by users' most preferred items from the entire corpus. In large-scale industrial recommender systems, making precise user preference prediction and efficient retrieval from a big corpus is extremely important. The linear retrieval complexity of traversing the entire corpus is usually unacceptable. Together with a user preference prediction model, a proper index structure is usually necessary in retrieving the candidate set.

Recently, vector representation learning methods have become more and more popular in recommender systems [2, 14, 17, 29]. In these methods, both users and items are firstly represented by vectors in a same space. Then the inner-product of the user vector and the item vector is used as the metric of user-item preference. As a main benefit, the candidate generation for these methods equals to a classic k-nearest neighbour problem, which can be accelerated by quantization-based index [10, 16], hierarchical graph index[18] etc.. Many efforts such as sequential model [17], graph convolutional network [29] have been made to learn better user vectors and item vectors. However, the inner-product form of user-item preference modeling required by these methods is still a bottleneck for improving user preference prediction accuracy[7, 39].

To break this bottleneck, a tree structure is used as index in tree-based methods [20, 38, 39, 41]. The general framework of tree-based methods for recommender systems is shown in Figure 1(a). Firstly, each item in the corpus is carefully indexed to a leaf node of $\mathcal{T}$ and a user node preference prediction model $\mathcal{M}$ is trained. $\mathcal{T}$ is usually a full binaray tree and the relationship between leaf nodes and items in corpus can be made by clustering [39] or joint learning [38]. In retrieval, a top-down beam search process on $\mathcal{T}$ guided by $\mathcal{M}$ is used in candidate generation, which complexity is logarithmic w.r.t. the corpus size. With tree index, restrictions on the structure of $\mathcal{M}$ required by the kNN-based retrieval are removed and many advanced user preference prediction model such as Deep Interest Network [37], Wide & Deep [1], Deep Interest Evolution Network

[*] Daqing Chang, Jintao Liu, Ziru Xu contributed equally to this work.

DLP-KDD 2021, August 15, 2021, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[36], xDeepFM [15] can be naturally used to achieve better user preference accuracy [39].

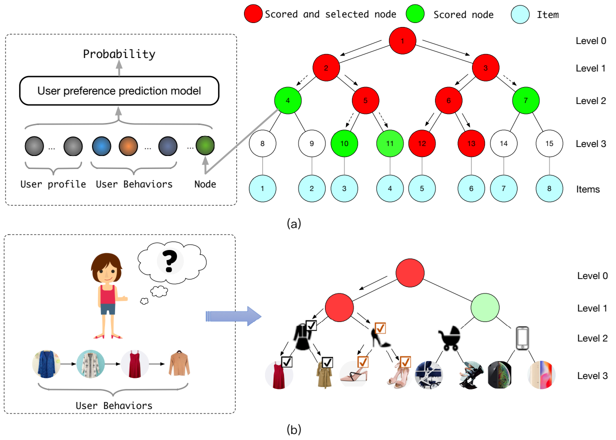

Figure 1: (a). A general framework of tree-based model for recommender systems. In retrieval, a top-down beam search on the tree index is firstly processed to generate the final candidate set guided by a user preference prediction model. Then items indexed on nodes in the beam of the leaf level are generated as candidate sets. In this example, the beam size is 2 and item 5, 6 are finally selected for recommendation. (b) A toy example to show the value of contexts between nodes on $\mathcal{T}$.

However, all models above are originally designed for general user preference prediction and the useful contexts between nodes on tree index are not fully considered. Since nodes with a common parent are equally represented by this parent both in training and retrieval, each node on $\mathcal{T}$ is actually an abstraction of its children. The beam search retrieval is actually to generate the final candidate set from coarse to detail. Intuitively, contexts between nodes on $\mathcal{T}$ should be useful auxiliary knowledge in user preference prediction. To better illustrate this, a toy example is shown in Fig 1(b). According to the girl's historical behaviors, she probably prefers cloths at this time. Suppose $\mathcal{M}$ has rightly predicted the left node in level 2 as the girl's coarse preference, this vertical context should be useful for $\mathcal{M}$ to generate the final candidate set in leaf level. Besides, relationships between nodes on the same level are also useful horizontal contexts in prediction. For this toy example, shoes are also probably preferred by the girl in Fig 1 because clothes and shoes are usually correlative things in daily life. However, shoes may not be chosen by $\mathcal{M}$ if only historical behaviors are used as features in prediction.

In this paper, we propose a Context-aware Tree-based Deep Model (ConTDM) to utilize both vertical and horizontal contexts on tree index in user preference prediction for tree-based recommender systems. Generally, a novel context-aware user node preference prediction model $\mathcal{M}$ is proposed in ConTDM. Our main contributions are listed as follows:

- Horizontally, contexts between nodes on the same level of $\mathcal{T}$ are aggregated with a graph convolutional layer. For tree-based models, a hierarchical graph structure is necessary to utilize vertical contexts on all levels of $\mathcal{T}$. We propose a novel hierarchical graph construction algorithm according to raw user behavior sequences and tree index.
- Vertically, a parent fusion layer is designed in $\mathcal{M}$. In ConTDM, we take the user preference representations predicted on higher levels of $\mathcal{T}$ as vertical contexts. Through the parent fusion layer, they are imported as an auxiliary input in prediction on the current level.
- We argue that the proposed user preference model can be conveniently extended to other tree-based methods. The training of ConTDM in this paper follows the framework proposed in TDM [39] without loss of generality. Offline experiments and ablation study on open data sets shows the significant improvements of ConTDM compared with baseline methods.
- ConTDM has been applied in full production to the display advertising scenario of Guess What You Like column of Taobao App Homepage at the candidate generation stage. Online A/B test shows the significant improvements on click-through rate (CTR) and revenue per mille (RPM), which are key performance indicators for online display advertising.

The rest of the paper is organized as follows: We introduce related works in Section 2. The proposed context-aware user preference prediction model and the training framework of ConTDM are introduced in Section 3. Experimental results are analysed in Section 4. We conclude our work in Section 5.

2 RELATED WORK

Vector representation learning methods beginning from matrix factorization based collaborative filtering have been widely used in recommender systems [2, 13, 14, 27, 29, 32]. In these methods, both users and items are mapped to vectors in the same space and the user-item preference is measured by the inner-product of user and item vectors. As an early representative work, a multi-layer fully connected network is used to project users and items into a latent space in industrial YouTube video recommender systems [2]. As a main benefit, the candidate retrieval for these methods equals to a k-nearest neighbour problem, which can be accelerated by quantization based index [10, 16], hierarchical graph index[18] etc.. Variants of improvements have been made in learning the vector mappings. For example, Lv et al. [17] use both recurrent neural network and attention mechanism in learning user vectors. Wang et al. [29] use a graph neural network to aggregate local information from the graph structure. However, the simple inner-product form of user preference modeling required by the kNN-based retrieval is still a key bottleneck for recommendation accuracy due to its limited learning capacity [7, 39]. Many other user preference models that has been shown to be effective such as Deep Interest Network [37], Deep & Wide [1], Deep Interest Evolution Network [36], xDeepFM [15] usually can not be applied directly in these methods.

In past years, tree-based methods are actively studied in the field of extreme classification [3, 20, 22, 23, 25, 30, 34], which is also closely related with recommender systems [8, 22]. To break the bottleneck of kNN-based methods, tree-based methods [38, 39, 41] take a tree structure as index and each item in corpus is attached to a leaf node by clustering [39] or joint learning [38]. A beam search

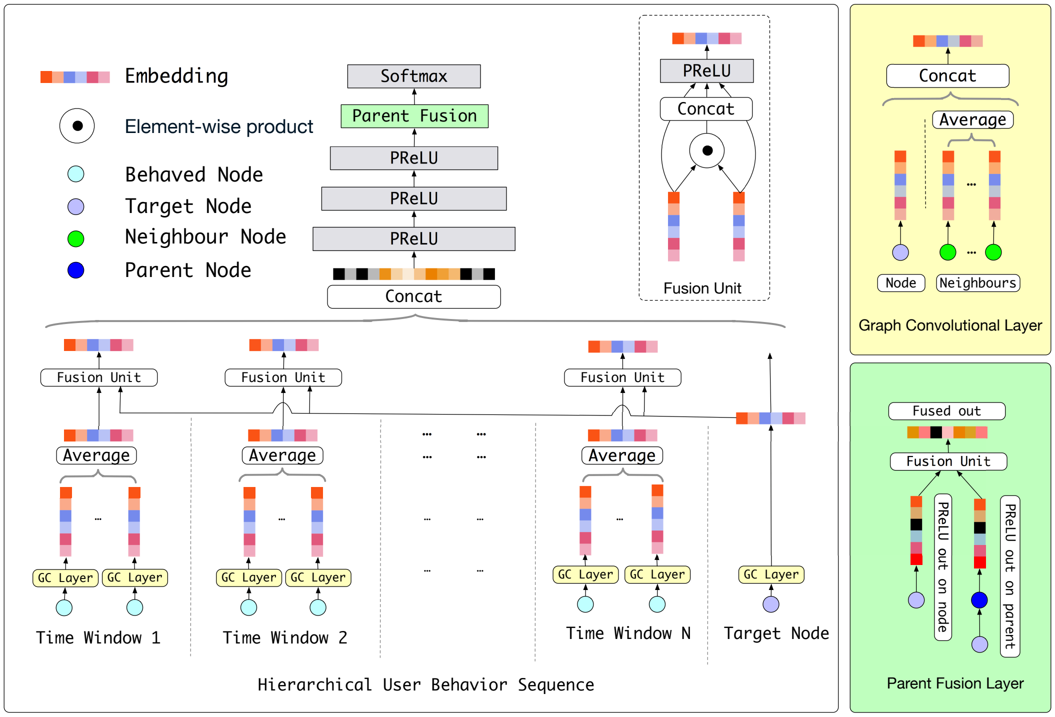

Figure 2: The backbone of user preference prediction model in ConTDM. Highlights: a) A graph convolutional layer is used to aggregate horizontal contexts on the tree index. Both the user embeddings and target node embedding are enhanced by this layer. b) The parent fusion layer takes both the output of the multi-layer fully connected network on the target node and its parent as inputs and a fusion unit is used to utilize vertical contexts on the tree index.

process from top to bottom is used in retrieval and a logarithmic complexity w.r.t. the corpus size is achieved. In these methods, restrictions on the form of user preference prediction model is removed and the use of arbitrary advanced user preference models is enabled to improve the recommendation accuracy. In training, a joint optimization framework of the user preference prediction model and the tree index [38] is proposed. More recently, An optimal beam search aware training framework of tree-based deep models [41] is proposed to eliminate the mismatch in training and retrieval.

On the other hand, graph-based methods have also attracted much attention in recommender systems [4, 28, 31, 33, 35, 40]. The general idea of graph-based methods is to make effective information aggregation from the local subgraph with a well constructed graph structure. Variants of aggregator architectures have been proposed in literatures [5, 12, 26]. In industrial community, Ying et al. [33] propose the PinSage algorithm used in Pinterest by adding a random walk based neighbour sampling strategy to GraphSage [5]. By importing side information to DeepWalk [21], Wang et al. [28] propose the graph embedding algorithm used in Alibaba to generate the item embeddings. They are used to calculate the similarity matrix for subsequent Item-CF [24] based recommendation. However, to our best knowledge, there is no existing work applied in industrial community simultaneously utilizing the superiority of both tree-based methods and graph-based models.

3 CONTEXT-AWARE TREE-BASED DEEP MODEL

In this section, we introduce the proposed Context-aware Tree-based Deep Model for recommender systems. As a main highlight of ConTDM, the proposed context-aware user node preference prediction model is given in Section 3.1. We show the training framework of ConTDM in Section 3.2.

3.1 Context-aware User Preference Prediction Model

In tree-based methods for recommendation, a tree structure $\mathcal{T}$ is used as index and each item in corpus is carefully indexed to a leaf node on $\mathcal{T}$ by clustering [39], joint learning [38] etc., as introduced in Section 1 and Fig 1. Then a beam search process guided by a user node preference prediction model $\mathcal{M}$ is made to generate the candidate set for recommendation in retrieval. Obviously, the accuracy of user node preference prediction has great influence on the final recommendation quality for tree-based methods. Inspired by the essence of tree-based methods is to generate the candidate set from coarse to detail, we hope to fully utilize the rich hierarchical information on the tree index about the corpus in designing the structure of $\mathcal{M}$. More specifically, both vertical and horizontal contexts contained on $\mathcal{T}$ are properly utilized to improve the accuracy of $\mathcal{M}$ in ConTDM. The backbone of $\mathcal{M}$ is shown in Fig 2.

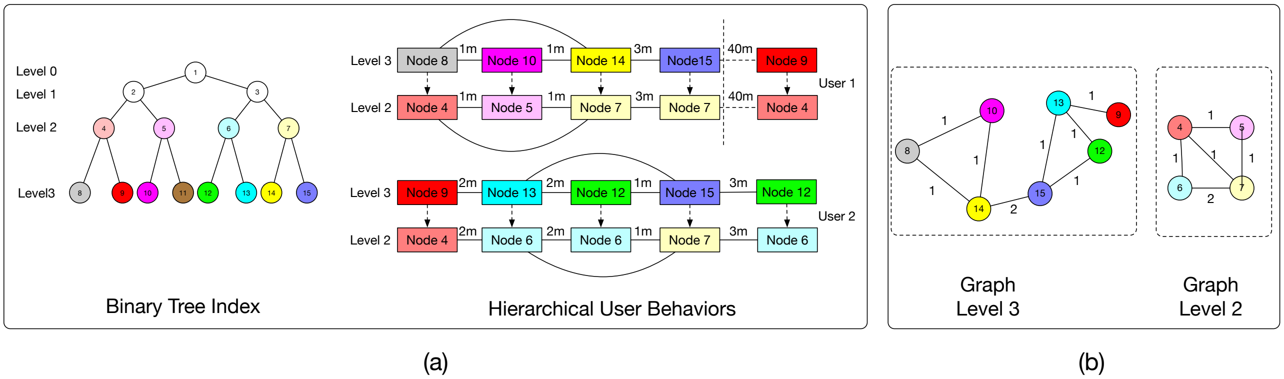

Figure 3: (a) An example to generate edges based on tree index and raw user behaviors. Firstly, hierarchical behavior sequences are contructed. Then an edge is put into the graph between an co-occurrence pair of nodes if the gap of their time stamps is not bigger than a threshold. (b) The constructed hierarchical graph based on sequences in (a). The weight of an edge is the count of co-occurrence between its two endpoints.

Denote $(c_1, c_2 \cdots c_m)$ as the historical behavior sequence of the user u and denote $b_j(c)$ as the ancestor of node c on level j of $\mathcal{T}$. We use hierarchical user behavior sequences [38] that have been shown effective in tree-based methods for recommender systems as our user features. More exactly, the user feature is represented as $(b_j(c_1), b_j(c_2) \cdots\ b_j(c_m))$ when the target node in prediction is on level j of $\mathcal{T}$. Other useful features such as user profiles can be conveniently added if needed. In estimating the user node preference probability, the embeddings of both the hierarchical user behavior sequence and the target node are firstly put into a graph convolutional layer to import horizontal contexts on $\mathcal{T}$. Then user historical behaviors are divided into different time windows and graph embeddings in the same time window are averaged to reduce the network complexity, which is optional to meet the practical time constraint in large scale industrial application without much loss of the effect at the same time. Next, a fusion unit is used to fuse the graph embeddings in user behaviors with the graph embedding of the target node similarly as the attention mechanism, which can also been employed in practice. We use the fusion unit here for consistency. After this, the user embeddings and the target embedding are concatenated as the input of a multi-layer fully connected network with PReLU activation function. The outputs of the network on both the target node and its parent are further fused by a parent fusion layer to import vertical contexts. Finally, a softmax layer is used to compute the user-node preference probability.

As key components to utilize contexts between nodes, the mechanism and effect of the graph convolutional layer and the parent fusion layer are further analysed in the following two subsections.

3.1.1 *Graph Convolutional Layer*.

Horizontally, the relevance between nodes on the same level of $\mathcal{T}$ can be used to enrich both the user and the target node features. As discussed in Section 2, graph structures have been proven to be powerful to grep the relationship between items in corpus.

In ConTDM, a non-parameterized graph convolutional layer based on GraphSage[5] is used considering the efficiency in training and inference, as shown in Fig 2. For each node n on the tree index, its graph embedding is computed by:

$$Emb_{ge}(n) = Concat(Emb(n), Avg(Emb(Neigh(n)))) \tag{1}$$

where $Emb(n)$ is the embedding of node n and $Neigh(n)$ is the set of $n's$ neighbours on the graph. The graph convolutional layer concatenates the averaged embeddings of $n's$ neighbours with $n's$ embedding. The purpose is to import the neighbour context as well as highlight the role of n.

With the graph convolutional layer, the representation of each node is enriched by its neighbours. On the one hand, the impact of sparsity in training data that has been widely observed in recommender systems [24] is alleviated. On the other hand, the scope of the user feature could jump out of historical behaviors and the diversity of the candidate set is promoted, which is usually preferred by most recommender systems. Besides, it is worth to point out that graph embeddings of all nodes can be computed efficiently before the trained model is put online according to the backbone of $\mathcal{M}$. Therefore, as another benefit brought by the graph convolutional layer, the time cost of online forward computation which is usually strictly bounded in industrial recommendation scenario can be saved.

Obviously, the quality of the graph matters. Besides, a hierarchical graph is required in ConTDM to aggregate horizontal contexts on all levels of $\mathcal{T}$. We build the hierarchical graph according to the co-occurrence of nodes in hierarchical user behavior sequences $\{(b_j(c_1), b_j(c_2) \cdots\ b_j(c_m))\}$. The process is explained in Fig 3(a). The user behavior sequence is firstly divided into sessions according to the gap between timestamps. Then an edge between each co-occurrence pair in the same session is added to the graph. The weight of an edge is the count of all co-occurrence pairs between its two endpoints. Generally, our main idea is that two items behaved by a user sequentially are probably related with each other and the hierarchical behavior sequence is the abstract user behaviors on different levels of $\mathcal{T}$.A formal statement about the construction of the hierarchical graph is given in Alg 1 where t is the threshhold (e.g. $40min$) to divide the sessions in behavior sequences and k is the max number of neighbours used in truncation (e.g. 10 in practice)

Algorithm 1 Building Hierarchical Graph in ConTDM

```
Input: Tree index $\mathcal{T}$ with max level $l_{max}$, raw user behavior sequences of training data $\{(c_1, c_2 \cdots c_m)\}$, max number of neighbours $k$, max time interval $t$
    $\mathcal{G} \leftarrow \emptyset$
for $l = l_{max}, \ldots, 1$ do
    for each raw sequence $(c_1, c_2 \cdots c_m)$ do
        $(b_l(c_1), b_l(c_2) \cdots b_l(c_m)) \leftarrow$ Trace each item in raw sequence up to level $l$ of $\mathcal{T}$.
        if node $n_i$ and $n_j$ are both contained in $(b_l(c_1), b_l(c_2) \cdots b_l(c_m))$ and the time interval between them $\leq t$ minutes then
            Increase the weight of undirected edge $(n_i, n_j)$ in $\mathcal{G}$ by 1.
        end if
    end for
    for each node $n$ in level $l$ of $\mathcal{T}$ do
        $\{n_i\}_{i=1}^{K} \leftarrow$ Pick the other endpoint of all undirected edges containing $n$ in $\mathcal{G}$.
        Sort $\{n\}_{i=1}^{K}$ in the descending order of $(n, n_i)$'s weight.
        for $i = 1, \ldots, \min(k, K)$ do
            if edge $(n, n_i)$ not in $\mathcal{G}$ then
                Add undirected edge $(n, n_i)$ to $\mathcal{G}$.
            end if
        end for
    end for
end for
Output: Undirected hierarchical graph $\mathcal{G}$.
```

to avoid the number of neighbours for hot nodes is too big and only keep the solid relationship as well.

3.1.2 *Parent Fusion Layer.*

Given a user u and a target node n_l on level l of $\mathcal{T}$, we denote $pa(n_l)$ as the parent node of n_l in level $l-1$ and denote $v(n_l)$ as the output of the multi-layer fully connected network. The parent fusion layer takes $v(n_l)$ and $v(pa(n_l))$ as input and returns the fused user node preference representation through a fusion unit. The detailed formulation is

$$v_f = PReLU\left(W * Concat\left(v(n_l), v(n_l) \odot v(pa(n_l)), v(pa(n_l))\right) + b\right). \tag{2}$$

where W is the weight term and b is the bias term to be optimized. With the parent fusion layer, the vertical contexts on higher levels of $\mathcal{T}$ are imported to the current prediction. The impact of the parent fusion layer is shown as follows:

- **Explainability**. In the tree index, each node is an abstraction of its children. Traditionally, $v(n_l)$ is used as the input vector of the final softmax layer to compute the final user preference probability, which indicates that $v(n_l)$ should contain useful user preference information. Besides, the essence of the hierarchical retrieval is to generate the final candidate set from coarse to fine. Therefore, the user preference representation on the parent node $v(pa(n_l))$ is a useful coarse-grained auxiliary feature in the prediction of n_l.
- **Non-sparsity**. In ConTDM, the training samples for each node in higher levels is much more enriched than lower levels, since the total number of nodes decreases exponentially with the going up of levels on $\mathcal{T}$. By importing contexts in higher levels, the impact of sparsity in lower levels is also largely reduced.
- **Efficiency**. Since the hierarchical retrieval is made from top to bottom on $\mathcal{T}$, $v(pa(n_l))$ can be efficiently reused in predicting n_l. Therefore, the total increase of computation in retrieval is brought by the fusion unit, which is usually acceptable.

Algorithm 2 The Training Framework of ConTDM

```
Input: Initial context-aware user preference model $\mathcal{M}$, tree index $\mathcal{T}$, raw training data $\{(u_i, q_i)\}_{i=1}^{N}$.
Construct the hierarchical graph including all items in corpus and nodes on $\mathcal{T}$ as vertex with Alg 1.
for t=1,2$\cdots$ T do
    Construct training samples used in current iteration.
    Optimize $\mathcal{M}$ using algorithms such as ADAM [11].
end for
Output: Trained model $\mathcal{M}$ used for beam search retrieval.
```

3.2 Training Framework

With the proposed context-aware user preference prediction model, the training framework of ConTDM is shown in Alg 2 following the tree-based deep model proposed in [39].

The input of Alg 2 contains an initial context-aware user preference model $\mathcal{M}$, the tree index $\mathcal{T}$ and the raw training data set $\{(u_i, q_i)\}_{i=1}^{N}$ where u_i denotes user i and q_i denotes the label item u_i prefers (e.g. u_i clicks q_i before). The initial tree index can be constructed by clustering following [39] without loss of generality. Before training $\mathcal{M}$, we firstly construct the hierarchical graph $\mathcal{G}$ used by Alg 1. Next, $\mathcal{M}$ is optimized under the total empirical loss as follows [39]:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \sum_{j=0}^{l_{max}} \log \hat{p}\left(b_j(q_i)|u_i; \boldsymbol{\theta}\right) \tag{3}$$

where l_{max} is the max level of tree index $\mathcal{T}$. $\boldsymbol{\theta}$ is the parameter of $\mathcal{M}$ to be optimized. $b_j(q)$ returns item $q's$ ancestor node on level j of $\mathcal{T}$. $\hat{p}(q|u)$ is the estimated probability u prefers q by $\mathcal{M}$. In Eq (3), the total negative logarithm of the estimated user-node probability between each pair (u_i, q_i) and their ancestors is minimized. In each iteration, we randomly sample a mini-batch samples from the raw data set and tracing them up to all levels of $\mathcal{T}$ as positive data. Besides, negative sampling [2, 9] is used in estimating $\hat{p}(q|u)$ with negative data sampled from the corresponding levels of $\mathcal{T}$.

As the proposed context-aware user node preference prediction model in ConTDM only relies on the tree index and raw training behavior sequences, it is convenient to be applied to other existing tree-based frameworks for recommender systems such as [20, 38, 41].

4 EXPERIMENTS

In this section, we show both online and offline performance of ConTDM. Firstly, datasets utilized in offline experiments are briefly summarized. Secondly, we compare the overall performance of ConTDM with other baseline recommendation models to show the effectiveness of the context-aware modeling. Thirdly, ablation study is followed up to help comprehend how each part of ConTDM works in detail. At last, we show the performance of ConTDM in Taobao display advertising platform with real online traffic.

Our offline experiments are conducted with two large-scale real-world datasets: 1) user-book review dataset from Amazon[6, 19]; 2) user-item behavior dataset from Taobao called UserBehavior[39]. The details are as follows:

- **Amazon Books**: This dataset is composed by product reviews from Amazon. Here we use its largest subset, i.e., `Books`. The users with less than 10 books reviewed are excluded. Each review record is in the format of `(user ID, book ID, rating, timestamp)`.
- **UserBehavior**: It is a subset of Taobao user behavior data containing about 1 million randomly sampled users who had behaviors from November 25 to December 03, 2017. Similar to Amazon Books, only users with at least 10 behaviors are kept. Each user-item behavior is corresponding to a record in the form of `(user ID, item ID, category ID, behavior type, timestamp)`. All behavior types are treated equal in our experiments.

Table 1 summarizes the above two datasets after preprocessing.

Table 1: Details of the two datasets after preprocessing. One record is a user-item pair that represents user feedback.

	Amazon Books	UserBehavior
# of users	294,739	969,529
# of items	1,477,922	4,162,024
# of categories	2,637	9,439
# of records	8,654,619	100,020,395

4.1 Experiment Setup

In our offline experiments, Precision, Recall and F-Measure are used as metrics for performance evaluation of different methods as in most related works for candidate generation in recommender systems. For a user u, denote $\mathcal{P}_u(|\mathcal{P}_u| = M)$ as the recalled candidate set and $\mathcal{G}_u$ as the ground truth set. The definitions of these metrics are as follows:

$$\text{Precision@}M(u) = \frac{|\mathcal{P}_u \cap \mathcal{G}_u|}{|\mathcal{P}_u|},\ \text{Recall@}M(u) = \frac{|\mathcal{P}_u \cap \mathcal{G}_u|}{|\mathcal{G}_u|},$$

$$\text{F-Measure@}M(u) = \frac{2 * \text{Precision@}M(u) * \text{Recall@}M(u)}{\text{Precision@}M(u) + \text{Recall@}M(u)}.$$

The user average of the above three metrics in testing set are used to compare the following methods:

- **Item-CF** [24], namely the classic item-based collaborative filtering, maintains an item-item matrix measuring similarities between pairs of items. The recommended items are generated according to the user's historical behaviors and the matrix.
- **YouTube product-DNN** [2], the representative work of kNN-based methods, is a practical method used in YouTube video recommendation. The inner-product of the learnt user and item's vector representation denotes the preference.
- **HSM** [20] is short for the hierarchical softmax model, which utilize multiplication of level-wise conditional probabilities to obtain item preference probability without the normalization term.
- **TDM** [39] is a representative tree-based deep model for recommender systems. The backbone of its user preference prediction model is comprised by an attention layer and a multi-layer plain-DNN.
- **ConTDM** is the proposed context-aware user preference model along with the tree index. The structural information on the tree index is incorporated by a graph convolutional layer and a parent fusion layer contained in the preference model.

We randomly sample 5,000 and 10,000 disjoint users to create testing set for Amazon Books and UserBehavior respectively. The other users in two datasets compose the training set. For each user in testing set, we take the first half of behaviors along the time line as known features and the latter half as ground truth. In negative samples generation, we deploy the same sampling strategy for all methods except Item-CF and use the same sampling ratio. For fairness, both HSM and TDM use the same user preference prediction model, which contains an attention layer before a three-layer plain-DNN. In ConTDM, the parameter size of the fusion unit is taken as closely as the attention layer in TDM and the plain-DNN layers are the same as other baseline methods. Note that we do not apply attention module to YouTube product-DNN because pairwise attention is not applicable in industrial scenario for user preference models with the inner-product form to achieve acceleration in retrieval. Besides, the same tree index learnt by the joint learning framework [38] with a plain-DNN user preference prediction model is shared by HSM, TDM and ConTDM to make fair comparison.

4.2 Comparison results

The quantitative results of all methods in two datasets under settings above is shown in Table 2.

Firstly, compared with the traditional Item-CF and kNN-based retrieval models, TDM significantly improves the recommendation accuracy in all metrics. This result clearly shows the superiority of tree-based methods by removing the restrictions on the form of user preference modeling and enabling the use of more effective models. Actually, together with a carefully designed joint learning framework[38], TDM could outperform the brutal-force traverse of the whole corpus with the same preference model trained on raw training data only. Besides, a direct application of hierarchical softmax model to recommendation problem does not show much improvements on Item-CF and knn based method, which is consistent with the conclusion in [39].

Secondly, ConTDM still outperforms the strongest baseline TDM with a 6.3% and 10.5% recall lift in Amazon Books and UserBehavior respectively. The comparison result shows the effectiveness of

Table 2: Comparison results of different methods in Amazon Books and UserBehavior.

Method	Amazon Books			UserBehavior		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Item-CF	0.52%	8.18%	0.92%	1.56%	6.75%	2.30%
YouTube product-DNN	0.53%	8.26%	0.93%	2.25%	10.15%	3.36%
HSM	0.57%	8.68%	1.00%	2.22%	10.42%	3.34%
TDM	0.83%	13.56%	1.49%	3.26%	15.50%	4.92%
ConTDM	**0.87%**	**14.42%**	**1.55%**	**3.65%**	**17.12%**	**5.49%**

importing contexts contained on the tree index to user preference modeling. Notice that the improvements are mostly achieved by the proposed context-aware user preference prediction model with a common tree index shared between different tree-based methods. A fine-tuned tree index and more effective training framework such as [38, 41] can be naturally applied in ConTDM to achieve better overall performance.

4.3 Ablation analysis

In ConTDM, a graph convolutional layer and a parent fusion layer are designed in the user preference prediction model to utilize both horizontal and vertical contexts on $\mathcal{T}$ respectively. In this subsection, we make an ablation analysis about the effectiveness of these two layers. All settings are kept unchanged as the last subsection other than removing one of these two layers from $\mathcal{M}$. Experimental results are shown in Table 3 with TDM as the baseline.

Graph Convolutional Layer. In this case, we remove the parent fusion layer in ConTDM and keep other parts unchanged. We denote this model as **ConTDM-GC**. From Table 3, ConTDM-GC lifts the recall with the relative percentage of 6.3% and 5.9% in Amazon Books and UserBehavior respectively. This result confirms the advantage of utilizing the co-occurrence among nodes from the same level in the tree index.

Parent Fusion Layer. In retrieval, the top-down path from the root to the leaf layer forms the decision chain of the user preference model, which naturally indicates the user interests granularity evolves from coarse to fine. From a probabilistic perspective, the top-down beam search process can be regarded as a sequence generation process. We remove the graph convolutional layer from ConTDM and denote the preference model with the parent fusion layer as **ConTDM-PF**. Results in Table 3 shows the effectiveness of the parent fusion layer. In UserBehavior, ConTDM-PF gains 6.6% recall lift, which beats the ConTDM-GC with 5.9%. Nevertheless, in Amazon Books, the recall yields by ConTDM-PF and TDM are roughly the same. We attribute this result to the impact of dataset. Since most items in this dataset are books and the coarse description of user preference on higher levels of $\mathcal{T}$ does not benefit much to prediction on child nodes.

Table 3: Ablation results for Graph Convolutional Layer and Parent Fusion Layer in Amazon Books and UserBehavior.

Dataset	Method	Metric@200		
		Precision	Recall	F-measure
Amazon Books	TDM	0.83%	13.56%	1.49%
	ConTDM-GC	0.88%	14.41%	1.57%
	ConTDM-PF	0.82%	13.55%	1.47%
	ConTDM	**0.87%**	**14.42%**	**1.55%**
UserBehavior	TDM	3.26%	15.50%	4.92%
	ConTDM-GC	3.50%	16.41%	5.26%
	ConTDM-PF	3.53%	16.53%	5.31%
	ConTDM	**3.65%**	**17.12%**	**5.49%**

4.4 Online Results

ConTDM has been applied in the display advertising scenario, i.e. Guess What You Like column of Taobao App, with full online traffic at the stage of candidate generation. In Taobao's advertising systems, advertisers bid on the reveals that show items to users. When a user opens the Taobao App, the advertising engine should choose a few proper ads from the large scale corpus of ads to be revealed. Generally, the whole process in practice can be devided into three subsequent stages: candidate generation, ranking and strategy. After each stage, the candidate set is reduced gradually from the whole corpus containing millions of items to few ads. Besides, the target considered in each stage also varies to meet different business goals.

To measure the performance, we conduct online A/B comparison by replacing ConTDM with the strongest baseline method TDM. Each comparison experiment has 2% of all online traffic. We use click-through rate (CTR) and revenue per mille (RPM) that are the key performance indicators for online display advertising as metrics. The definitions of these two metrics are as follows:

$$\text{CTR} = \frac{\text{\# of clicks}}{\text{\# of impressions}}, \ \text{RPM} = \frac{\text{Ad revenue}}{\text{\# of impressions}} * 1000.$$

Besides, the diversity defined by the size of different categories in the candidate set is also considered.

Table 4: Online results in *Guess What You Like* column of Taobao App Homepage.

Metric	CTR	RPM	Diversity
ConTDM	+3.8%	+4.0%	+14.0%

Table 4 reveals the lift on all online metrics. 3.8% growth on CTR exhibits that more precise items have been recommended. RPM with the increase of 4.0% proves ConTDM can bring more income for Taobao advertising platform. Thanks to the horizontal contexts brought by the hierarchical graph, the diversity of candidate set is also significantly improved by 14.0%. It means more potential improvements can be made by subsequent stages since their corpus is largely enriched. Notice that there are sevaral different methods working simultaneously online at the candidate generation stage and ConTDM is only one of them. Besides, the cancandidate set returned in the candidate generation stage will be reranked by the subsequent stages to pick the few final ads. Therefore, the improvements achieved by ConTDM is very significant. Besides, as discussed in Section 3.1.1, graph embeddings of ConTDM are aggregated offline and all improvements compared with TDM are achieved without increase of the stress of the online engine.

5 CONCLUSION

In this paper, we study the effect of the tree index in user preference modeling and propose a context-aware user preference prediction model, which can be conveniently applied in general tree-based methods for recommender systems. Both horizontal and vertical contexts on $\mathcal{T}$ are utilized through a novel graph convolutional layer and a parent fusion layer. Both online and offline results show the significant improvements brought by ConTDM.

In ConTDM, we mainly focus on how to improve the recommendation accuracy of user preference model in tree-based methods. Actually, the quality of the tree index also matters [38]. Intuitively, we can naturally extend the binary tree index used in ConTDM to a multi-path tree index with the constructed hierarchical graph and its capacity is much increased. We will further study how to make effective and efficient training and retrieval on the more challenging multi-path tree index in our future work.

REFERENCES

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[2] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[3] Hal Daumé III, Nikos Karampatziakis, John Langford, and Paul Mineiro. 2017. Logarithmic time one-against-some. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 923–932.

[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[6] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[8] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 935–944.

[9] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2019).

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[13] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[15] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[16] Ting Liu, Andrew W Moore, Ke Yang, and Alexander G Gray. 2005. An investigation of practical approximate nearest neighbor algorithms. In *Advances in neural information processing systems*. 825–832.

[17] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2635–2643.

[18] Yury A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52.

[20] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model.. In *Aistats*, Vol. 5. Citeseer, 246–252.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[22] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*. 993–1002.

[23] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 263–272.

[24] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[25] Ryutaro Tanno, Kai Arulkumaran, Daniel C Alexander, Antonio Criminisi, and Aditya Nori. 2018. Adaptive neural trees. *arXiv preprint arXiv:1807.06699* (2018).

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

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

[28] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 839–848.

[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[30] Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label partitioning for sublinear ranking. In *International conference on machine learning*. 181–189.

[31] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

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

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

[34] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *Advances in Neural Information Processing Systems*. 5812–5822.

[35] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 635–644.

[36] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[38] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. 2019. Joint optimization of tree-based index and deep model for recommender systems. In *Advances in Neural Information Processing Systems*. 3973–3982.

[39] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1079–1088.

[40] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. AliGraph: a comprehensive graph neural network platform. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2094–2105.

[41] Jingwei Zhuo, Ziru Xu, Wei Dai, Han Zhu, Han Li, Jian Xu, and Kun Gai. 2020. Learning Optimal Tree Models under Beam Search. *arXiv preprint arXiv:2006.15408* (2020).