Learning Optimal Tree Models under Beam Search

Jingwei Zhuo [1] **Ziru Xu** [1] **Wei Dai** [1] **Han Zhu** [1] **Han Li** [1] **Jian Xu** [1] **Kun Gai** [1]

Abstract

Retrieving relevant targets from an extremely large target set under computational limits is a common challenge for information retrieval and recommendation systems. Tree models, which formulate targets as leaves of a tree with trainable node-wise scorers, have attracted a lot of interests in tackling this challenge due to their logarithmic computational complexity in both training and testing. Tree-based deep models (TDMs) and probabilistic label trees (PLTs) are two representative kinds of them. Though achieving many practical successes, existing tree models suffer from the training-testing discrepancy, where the retrieval performance deterioration caused by beam search in testing is not considered in training. This leads to an intrinsic gap between the most relevant targets and those retrieved by beam search with even the optimally trained node-wise scorers. We take a first step towards understanding and analyzing this problem theoretically, and develop the concept of *Bayes optimality under beam search* and *calibration under beam search* as general analyzing tools for this purpose. Moreover, to eliminate the discrepancy, we propose a novel algorithm for learning optimal tree models under beam search. Experiments on both synthetic and real data verify the rationality of our theoretical analysis and demonstrate the superiority of our algorithm compared to state-of-the-art methods.

1. Introduction

Extremely large-scale retrieval problems prevail in modern industrial applications of information retrieval and recommendation systems. For example, in online advertising systems, several advertisements need to be retrieved from a target set containing tens of millions of advertisements and presented to a user in tens of milliseconds. The limits of computational resources and response time make models, whose computational complexity scales linearly with the size of target set, become unacceptable in practice.

Tree models are of special interest to solve these problems because of their ability in achieving logarithmic complexity in both training and testing. Tree-based deep models (TDMs) (Zhu et al., 2018; 2019; You et al., 2019) and Probabilistic label trees (PLTs) (Jasinska et al., 2016; Prabhu et al., 2018; Wydmuch et al., 2018) are two representative kinds of tree models. These models introduce a tree hierarchy in which each leaf node corresponds to a target and each non-leaf node defines a pseudo target for measuring the existence of relevant targets on the subtree rooted at it. Each node is also associated with a node-wise scorer which is trained to estimate the probability that the corresponding (pseudo) target is relevant. To achieve logarithmic training complexity, a subsampling method is leveraged to select logarithmic number of nodes on which the scorers are trained for each training instance. In testing, beam search is usually used to retrieve relevant targets in logarithmic complexity.

As a greedy method, beam search only expands parts of nodes with larger scores while pruning other nodes. This character achieves logarithmic computational complexity but may result in deteriorating retrieval performance if ancestor nodes of the most relevant targets are pruned. An ideal tree model should guarantee no performance deterioration when its node-wise scorers are leveraged for beam search. However, existing tree models ignore this and treat training as a separated task to testing: (1) Node-wise scorers are trained as probability estimators of pseudo targets which are not designed for optimal retrieval; (2) They are also trained on subsampled nodes which are different to those queried by beam search in testing. Such discrepancy makes even the optimal node-wise scorers w.r.t. training loss can lead to suboptimal retrieval results when they are used in testing to retrieve relevant targets via beam search. To the best of our knowledge, there is little work discussing this problem either theoretically or experimentally.

We take a first step towards understanding and resolving the training-testing discrepancy on tree models. To analyze this formally, we develop the concept of *Bayes optimality under beam search* and *calibration under beam search* as

[1]Alibaba Group. Correspondence to: Jingwei Zhuo <zjw169463@alibaba-inc.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

the optimality measure of tree models and corresponding training loss, respectively. Both of them serve as general analyzing tools for tree models. Based on these concepts, we show that neither TDMs nor PLTs are optimal, and derive a sufficient condition for the existence of optimal tree models as well. We also propose a novel algorithm for learning such an optimal tree model. Our algorithm consists of a beam search aware subsampling method and an optimal retrieval based definition of pseudo targets, both of which resolve the training-testing discrepancy. Experiments on synthetic and real data not only verify the rationality of our newly proposed concepts in measuring the optimality of tree models, but also demonstrate the superiority of our algorithm compared to existing state-of-the-art methods.

2. Related Work

Tree Models: Research on tree models[1] has mainly focused on formulating node-wise scorers and the tree structure. For node-wise scorers, linear models are widely adopted (Jasinska et al., 2016; Wydmuch et al., 2018; Prabhu et al., 2018), while deep models (Zhu et al., 2018; 2019; You et al., 2019) become popular recently. For the tree structure, apart from the random tree (Jasinska et al., 2016), recent works propose to learn it either via hierarchical clustering over targets (Wydmuch et al., 2018; Prabhu et al., 2018; Khandagale et al., 2019) or under a joint optimization framework with node-wise scorers (Zhu et al., 2019). Without dependence on specific formulations of node-wise scorers or the tree structure, our theoretical findings and proposed training algorithm are general and applicable to these advances.

Bayes Optimality and Calibration: Bayes optimality and calibration have been extensively investigated on flat models (Lapin et al., 2017; Menon et al., 2019; Yang & Koyejo, 2019), and they have also been used to measure the performance of tree models on hierarchical probability estimation (Wydmuch et al., 2018). However, there is a gap between the performance on hierarchical probability estimation and that on retrieving relevant targets, since the former ignores beam search and corresponding performance deterioration. As a result, how to measure the retrieval performance of tree models formally remains an open question. We fill this void by developing the concept of Bayes optimality under beam search and calibration under beam search.

Beam Search in Training: Formulating beam search into training to resolve the training-testing discrepancy is not a new idea. It has been extensively investigated on structured prediction models for problems like machine translation and speech recognition (Daumé III & Marcu, 2005; Xu & Fern, 2007; Ross et al., 2011; Wiseman & Rush, 2016; Goyal et al., 2018; Negrinho et al., 2018). Though performance deterioration caused by beam search has been analyzed empirically (Cohen & Beck, 2019), it still lacks a theoretical understanding. Besides, little effort has been made to understand and resolve the training-testing discrepancy on tree models. We take a first step towards studying these problems both theoretically and experimentally.

[1]There also exist models which usually build an ensemble of decision trees over instances instead of targets (Prabhu & Varma, 2014; Jain et al., 2016). They are less relevant to our main focus.

3. Preliminaries

3.1. Problem Definition

Suppose $\mathcal{I} = \{1, ..., M\}$ with $M \gg 1$ is a target set and $\mathcal{X}$ is an observation space, we denote an instance[2] as $(\mathbf{x}, \mathcal{I}_\mathbf{x})$, implying that an observation $\mathbf{x} \in \mathcal{X}$ is associated with a subset of relevant targets $\mathcal{I}_\mathbf{x} \subset \mathcal{I}$, which usually satisfies $|\mathcal{I}_\mathbf{x}| \ll M$. For notation simplicity, we introduce a binary vector $\mathbf{y} \in \mathcal{Y} = \{0, 1\}^M$ as an alternative representation for $\mathcal{I}_\mathbf{x}$, where $y_j = 1$ implies $j \in \mathcal{I}_\mathbf{x}$ and vice versa. As a result, an instance can also be denoted as $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$.

Let $p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ be a probability density function for data which is unknown in practice, we slightly abuse notations by regarding an instance $(\mathbf{x}, \mathbf{y})$ as either the random variable pair w.r.t. $p(\mathbf{x}, \mathbf{y})$ or a sample of $p(\mathbf{x}, \mathbf{y})$. We also assume the training dataset $\mathcal{D}_{tr}$ and the testing dataset $\mathcal{D}_{te}$ to be the sets containing i.i.d. samples of $p(\mathbf{x}, \mathbf{y})$. Since $\mathbf{y}$ is a binary vector, we use the simplified notation $\eta_j(\mathbf{x}) = p(y_j = 1|\mathbf{x})$ for any $j \in \mathcal{I}$ in the rest of this paper.

Given these notations, the extremely large-scale retrieval problem is defined as to learn a model $\mathcal{M}$ such that its retrieved subset for any $\mathbf{x} \sim p(\mathbf{x})$, denoted by either $\hat{\mathcal{I}}_\mathbf{x}$ or $\hat{\mathbf{y}}$, is as close as $\mathbf{y} \sim p(\mathbf{y}|\mathbf{x})$ according to some performance metrics. Since $p(\mathbf{x}, \mathbf{y})$ is unknown in practice, such a model is usually learnt as an estimator of $p(\mathbf{y}|\mathbf{x})$ on $\mathcal{D}_{tr}$ and its retrieval performance is evaluated on $\mathcal{D}_{te}$.

3.2. Tree Models

Suppose $\mathcal{T}$ is a b-arity tree with height H, we regard the node at the 0-th level as the root and nodes at the H-th level as leaves. Formally, we denote the node set at h-th level as $\mathcal{N}_h$ and the node set of $\mathcal{T}$ as $\mathcal{N} = \bigcup_{h=0}^{H} \mathcal{N}_h$. For each node $n \in \mathcal{N}$, we denote its parent as $\rho(n) \in \mathcal{N}$, its children set as $\mathcal{C}(n) \subset \mathcal{N}$, the path from the root to it as $\mathrm{Path}(n)$, and the set of leaves on its subtree as $\mathcal{L}(n)$.

Tree models formulate the target set $\mathcal{I}$ as leaves of $\mathcal{T}$ through a bijective mapping $\pi : \mathcal{N}_H \to \mathcal{I}$, which implies $H = \mathrm{O}(\log_b M)$. For any instance $(\mathbf{x}, \mathbf{y})$, each node $n \in \mathcal{N}$ is defined with a pseudo target $z_n \in \{0, 1\}$ to measure the

[2]This summarizes many practical applications. For example, in recommendation systems, an instance corresponds to an interaction between users and items, where $\mathbf{x}$ denotes the user information and $\mathcal{I}_\mathbf{x}$ denotes the items in which the user are interested.

existence of relevant targets on the subtree of n, i.e.,

$$z_n = \mathbb{I}(\sum_{n' \in \mathcal{L}(n)} y_{\pi(n')} \geq 1), \tag{1}$$

which satisfies $z_n = y_{\pi(n)}$ for $n \in \mathcal{N}_H$.

By doing so, tree models transform the original problem of estimating $p(y_j|\mathbf{x})$ to a series of hierarchical subproblems of estimating $p(z_n|\mathbf{x})$ on $n \in \text{Path}(\pi^{-1}(j))$. They introduce the node-wise scorer $g : \mathcal{X} \times \mathcal{N} \to \mathbb{R}$ to build such a node-wise estimator for each $n \in \mathcal{N}$, which is denoted as $p_g(z_n|\mathbf{x})$ to distinguish from the unknown distribution $p(z_n|\mathbf{x})$. In the rest of this paper, we denote a tree model as $\mathcal{M}(\mathcal{T}, g)$ to highlight its dependence on $\mathcal{T}$ and g.

3.2.1. Training of Tree Models

The training loss of tree models can be written as $\text{argmin}_g \sum_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{tr}} L(\mathbf{y}, \mathbf{g}(\mathbf{x}))$, where

$$L(\mathbf{y}, \mathbf{g}(\mathbf{x})) = \sum_{h=1}^{H} \sum_{n \in \mathcal{S}_h(\mathbf{y})} \ell_{\text{BCE}}\left(z_n, g(\mathbf{x}, n)\right). \tag{2}$$

In Eq. (2), $\mathbf{g}(\mathbf{x})$ is a vectorized representation of $\{g(\mathbf{x}, n) : n \in \mathcal{N}\}$ (e.g., level-order traversal), $\ell_{\text{BCE}}(z, g) = -z \log(1 + \exp(-g)) - (1 - z) \log(1 + \exp(g))$ is the binary cross entropy loss and $\mathcal{S}_h(\mathbf{y}) \subset \mathcal{N}_h$ is the set of subsampled nodes at h-th level for an instance $(\mathbf{x}, \mathbf{y})$. Let $C = \max_h |\mathcal{S}_h(\mathbf{y})|$, the training complexity is $\text{O}(HbC)$ per instance, which is logarithmic to the target set size M.

As two representatives of tree models, PLTs and TDMs adopt different ways[3] to build p_g and $S_h(\mathbf{y})$.

PLTs: Since $p(z_n|\mathbf{x})$ can be decomposed as $p(z_n = 1|\mathbf{x}) = \prod_{n' \in \text{Path}(n)} p(z_{n'} = 1|z_{\rho(n')} = 1, \mathbf{x})$ according to Eq. (1), $p_g(z_n|\mathbf{x})$ is decomposed accordingly via $p_g(z_{n'}|z_{\rho(n')} = 1, \mathbf{x}) = 1/(1 + \exp(-(2z_{n'} - 1)g(\mathbf{x}, n')))$. As a result, only nodes with $z_{\rho(n)} = 1$ are trained, which produces $\mathcal{S}_h(\mathbf{y}) = \{n : z_{\rho(n)} = 1, n \in \mathcal{N}_h\}$.

TDMs: Unlike PLTs, $p(z_n|\mathbf{x})$ is estimated directly via $p_g(z_n|\mathbf{x}) = 1/(1 + \exp(-(2z_n - 1)g(\mathbf{x}, n)))$. Besides, the subsample set[4] is chosen as $\mathcal{S}_h(\mathbf{y}) = \mathcal{S}_h^+(\mathbf{y}) \bigcup \mathcal{S}_h^-(\mathbf{y})$ where $\mathcal{S}_h^+(\mathbf{y}) = \{n : z_n = 1, n \in \mathcal{N}_h\}$ and $\mathcal{S}_h^-(\mathbf{y})$ contains several random samples over $\mathcal{N}_h \setminus \mathcal{S}_h^+(\mathbf{y})$.

3.2.2. Testing of Tree Models

For any testing instance $(\mathbf{x}, \mathbf{y})$, let $\mathcal{B}_h(\mathbf{x})$ denote the node set at h-th level retrieved by beam search and $k = |\mathcal{B}_h(\mathbf{x})|$ denote the beam size, the beam search process is defined as

$$\mathcal{B}_h(\mathbf{x}) \in \underset{n \in \tilde{\mathcal{B}}_h(\mathbf{x})}{\text{argTopk}}\, p_g(z_n = 1|\mathbf{x}), \tag{3}$$

where $\tilde{\mathcal{B}}_h(\mathbf{x}) = \bigcup_{n' \in \mathcal{B}_{h-1}(\mathbf{x})} \mathcal{C}(n')$.

By applying Eq. (3) recursively until $h = H$, beam search retrieves the set containing k leaf nodes, denoted by $\mathcal{B}_H(\mathbf{x})$. Let $m \leq k$ denote the number of targets to be retrieved, the retrieved target subset can be denoted as

$$\hat{\mathcal{I}}_\mathbf{x} = \{\pi(n) : n \in \mathcal{B}_H^{(m)}(\mathbf{x})\}, \tag{4}$$

where $\mathcal{B}_H^{(m)}(\mathbf{x}) \in \text{argTopm}_{n \in \mathcal{B}_H(\mathbf{x})}\, p_g(z_n = 1|\mathbf{x})$ denote the subset of $\mathcal{B}_H(\mathbf{x})$ with top-m scored nodes according to $p_g(z_n = 1|\mathbf{x})$. Since Eq. (3) only traverses at most bk nodes and generating $\mathcal{B}_H(\mathbf{x})$ needs computing Eq. (3) for H times, the testing complexity is $\text{O}(Hbk)$ per instance, which is also logarithmic to M.

To evaluate the retrieval performance of $\mathcal{M}(\mathcal{T}, g)$ on the testing dataset D_{te}, Precision@m, Recall@m and F-measure@m are widely adopted. Following Zhu et al. (2018; 2019), we define[5] them as the average of Eq. (5), Eq. (6) and Eq. (7) over D_{te} respectively, where

$$\text{P@}m(\mathcal{M}; \mathbf{x}, \mathbf{y}) = \frac{1}{m} \sum\nolimits_{j \in \hat{\mathcal{I}}_\mathbf{x}} y_j, \tag{5}$$

$$\text{R@}m(\mathcal{M}; \mathbf{x}, \mathbf{y}) = \frac{1}{|\mathcal{I}_\mathbf{x}|} \sum\nolimits_{j \in \hat{\mathcal{I}}_\mathbf{x}} y_j, \tag{6}$$

and

$$\text{F@}m(\mathcal{M}; \mathbf{x}, \mathbf{y}) = \frac{2 \cdot \text{P@}m(\mathcal{M}; \mathbf{x}, \mathbf{y}) \cdot \text{R@}m(\mathcal{M}; \mathbf{x}, \mathbf{y})}{\text{P@}m(\mathcal{M}; \mathbf{x}, \mathbf{y}) + \text{R@}m(\mathcal{M}; \mathbf{x}, \mathbf{y})}. \tag{7}$$

4. Main Contributions

Our main contributions can be divided into three parts: (1) We highlight the existence of the training-testing discrepancy on tree models, and provide an intuitive explanation of its negative effects on retrieval performance; (2) We develop the concept of Bayes optimality under beam search and calibration under beam search to formalize this intuitive explanation; (3) We propose a novel algorithm for learning tree models that are Bayes optimal under beam search.

4.1. Understanding the Training-Testing Discrepancy on Tree Models

According to Eq. (2), the training of $g(\mathbf{x}, n)$ depends on two factors: the subsample set $\mathcal{S}_h(\mathbf{y})$ and the pseudo target z_n.

[3] Details can be found in the supplementary materials.

[4] Zhu et al. (2018) defines TDM with the constraint $|\mathcal{I}_\mathbf{x}| = 1$, we extend their definition by removing this constraint and refer TDM to such an extended definition in the rest of this paper.

[5] Unlike macro/micro F-measure, the average of Eq. (7) over D_{te} defines the instance-wise F-measure. Wu & Zhou (2017, Table 1) provides a thorough comparison for them.

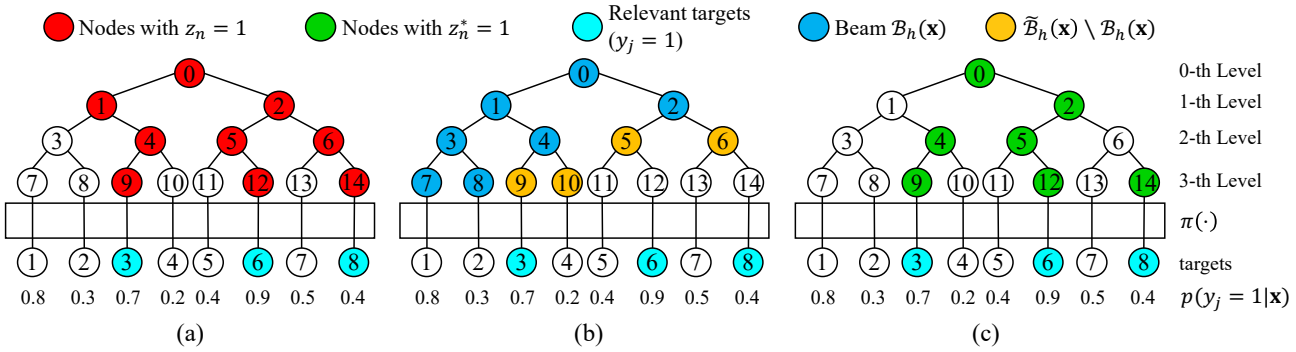

Figure 1. An overview of the training-testing discrepancy on a tree model $\mathcal{M}(\mathcal{T}, g)$. (a) The assignment of pseudo targets on existing tree models, where red nodes correspond to $z_n = 1$ defined in Eq. (1). (b) Beam search process, where targets mapping blue nodes at 3-th level (i.e., leaf nodes) are regarded as the retrieval results of $\mathcal{M}$. (c) The assignment of optimal pseudo targets based on the ground truth distribution $\eta_j(\mathbf{x}) = p(y_j = 1|\mathbf{x})$, where green nodes correspond to $z_n^* = 1$ defined in Eq. (13).

Table 1. Results for the toy experiment with $M = 1000$, $b = 2$. The reported number is $(\sum_{j\in\mathcal{I}^{(k)}} \eta_j - \sum_{j\in\hat{\mathcal{I}}} \eta_j)/k$, which is averaged over 100 runs with random initialization over $\mathcal{T}$ and η_j.

N	100	1000	10000	∞
$k = 1$	0.095	0.076	0.074	0.059
$k = 5$	0.075	0.055	0.050	0.037
$k = 10$	0.062	0.043	0.036	0.024
$k = 20$	0.057	0.036	0.031	0.018
$k = 50$	0.042	0.021	0.016	0.011

We can show that both factors relate to the training-testing discrepancy on existing tree models.

First, according to Eq. (3), the nodes at h-th level on which $g(\mathbf{x}, n)$ is queried in testing can be denoted as $\tilde{\mathcal{B}}_h(\mathbf{x})$, which implies a self-dependency of $g(\mathbf{x}, n)$, i.e., nodes on which $g(\mathbf{x}, n)$ is queried at h-th level depends on $g(\mathbf{x}, n)$ queried at $(h-1)$-th level. However, $\mathcal{S}_h(\mathbf{y})$, the nodes at h-th level on which $g(\mathbf{x}, n)$ is trained, is generated according to ground truth targets $\mathbf{y}$ via Eq. (1). Figure 1(a) and Figure 1(b) demonstrate such a difference: Node 7 and 8 (blue nodes) are traversed by beam search, but they are not in $\mathcal{S}_h(\mathbf{y})$ of PLTs and may not be in $\mathcal{S}_h(\mathbf{y})$ of TDMs according to $\mathcal{S}_h^+(\mathbf{y})$ (red nodes). As a result, $g(\mathbf{x}, n)$ is trained without considering such a self-dependency on itself when it is used for retrieving relevant targets via beam search. This discrepancy results in that $g(\mathbf{x}, n)$ trained well does not perform well in testing.

Second, z_n defined in Eq. (1) does not guarantee beam search w.r.t. $p_g(z_n = 1|\mathbf{x})$ has no performance deterioration, i.e., retrieving the most relevant targets. To see this, we design a toy example by ignoring $\mathbf{x}$ and defining the data distribution to be $p(\mathbf{y}) = \prod_{j=1}^{M} p(y_j)$, whose marginal probability $\eta_j = p(y_j = 1)$ is sampled from a uniform distribution in $[0, 1]$. As a result, we denote the training dataset as $\mathcal{D}_{tr} = \{\mathbf{y}^{(i)}\}_{i=1}^{N}$ and the pseudo target for instance $\mathbf{y}^{(i)}$ on node n as $z_n^{(i)}$. For $\mathcal{M}(\mathcal{T}, g)$, we assume $\mathcal{T}$ is randomly built and estimate $p(z_n = 1)$ directly via[6] $p_g(z_n = 1) = \sum_{i=1}^{N} z_n^{(i)}/N$ without the need to specify g, since there is no observation $\mathbf{x}$. Beam search with beam size k is applied on $\mathcal{M}$ to retrieve the target subset whose size is $m = k$ as well, denoted by $\hat{\mathcal{I}} = \{\pi(n) : n \in \mathcal{B}_H\}$. Since $p(\mathbf{y})$ is known in this toy example, we need no testing set $\mathcal{D}_{te}$ and evaluate the retrieval performance directly via the regret $(\sum_{j\in\mathcal{I}^{(k)}} \eta_j - \sum_{j\in\hat{\mathcal{I}}} \eta_j)/k$, where $\mathcal{I}^{(k)} \in \operatorname{argTopk}_{j\in\mathcal{I}} \eta_j$ denotes the top-k targets according to η_j. As a special case of Eq. (10), this metric quantifies the suboptimality of $\mathcal{M}$ and we'll discuss it formally later.

As is shown in Table 1, we can find that the regret is always non-zero with varying training data number N and beam size k. Even in the ideal case when $N = \infty$ and thus $p_g(z_n = 1) = p(z_n = 1)$, it is still non-zero. This implies that z_n defined in Eq. (1) cannot guarantee optimal retrieval performance in general. This phenomenon does not contradict with the zero regret property in Wydmuch et al. (2018, Theorem 2), since their theorem defines the regret using $\hat{\mathcal{I}} = \operatorname{argTopk}_{j\in\mathcal{I}} p_g(y_j = 1|\mathbf{x})$, which ignores the performance deterioration caused by beam search.

4.2. Bayes Optimality and Calibration under Beam Search

In Sec. 4.1, we discuss the existence of the training-testing discrepancy on tree models and provide a toy example to ex-

[6] Estimating $p(z_n = 1)$ hierarchically via $p_g(z_n = 1) = \prod_{n'\in\text{Path}(n)} p_g(z_{n'} = 1|z_{\rho(n')} = 1)$ and $p_g(z_n = 1|z_{\rho(n)} = 1) = \sum_{i=1}^{N} z_n^{(i)} z_{\rho(n)}^{(i)} / \sum_{i=1}^{N} z_{\rho(n)}^{(i)}$ provides similar results.

plain its effect. Without loss of generality, we formalize this discussion with Precision@m as the retrieval performance metric in this subsection.

The first question is, what does "optimal" mean for tree models with respect to their retrieval performance. In fact, the answer has been partially revealed by the toy example in Sec. 4.1, and we give a formal definition as follows:

Definition 1 (Bayes Optimality under Beam Search)**.** *Given the beam size k and the data distribution $p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, a tree model $\mathcal{M}(\mathcal{T}, g)$ is called top-k Bayes optimal under beam search if*

$$\{\pi(n) : n \in \mathcal{B}_H(\mathbf{x})\} \in \operatorname*{argTopk}_{j \in \mathcal{I}} \eta_j(\mathbf{x}), \tag{8}$$

holds for any $\mathbf{x} \in \mathcal{X}$. $\mathcal{M}(\mathcal{T}, g)$ is called Bayes optimal under beams search if Eq. (8) holds for any $\mathbf{x} \in \mathcal{X}$ and $1 \le k \le M$.

Given Definition 1, we can derive a sufficient condition for the existence of such an optimal tree model as follows[7] :

Proposition 1 (Sufficient Condition for Bayes Optimality under Beam Search)**.** *Given the beam size k, the data distribution $p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, the tree $\mathcal{T}$ and*

$$p^*(z_n|\mathbf{x}) = \begin{cases} \max\limits_{n' \in \mathcal{L}(n)} \eta_{\pi(n')}(\mathbf{x}), & z_n = 1 \\ 1 - \max\limits_{n' \in \mathcal{L}(n)} \eta_{\pi(n')}(\mathbf{x}), & z_n = 0 \end{cases}, \tag{9}$$

a tree model $\mathcal{M}(\mathcal{T}, g)$ is top-m Bayes optimal under beam search for any $m \le k$, if $p_g(z_n|\mathbf{x}) = p^*(z_n|\mathbf{x})$ holds for any $\mathbf{x} \in \mathcal{X}$ and $n \in \bigcup_{h=1}^{H} \tilde{\mathcal{B}}_h(\mathbf{x})$. $\mathcal{M}(\mathcal{T}, g)$ is Bayes optimal under beam search, if $p_g(z_n|\mathbf{x}) = p^*(z_n|\mathbf{x})$ holds for any $\mathbf{x} \in \mathcal{X}$ and $n \in \mathcal{N}$.

Proposition 1 shows one case of what an optimal tree model should be, but it does not resolve all the problems, since both learning and evaluating a tree model require a quantitative measure of its suboptimality. Notice that Eq. (8) implies that $\mathbb{E}_{p(\mathbf{x})}\left[\sum_{j \in \mathcal{I}_{\mathbf{x}}^{(k)}} \eta_j(\mathbf{x})\right] = \mathbb{E}_{p(\mathbf{x})}\left[\sum_{n \in \mathcal{B}_H(\mathbf{x})} \eta_{\pi(n)}(\mathbf{x})\right]$, where $\mathcal{I}_{\mathbf{x}}^{(k)} = \operatorname{argTopk}_{j \in \mathcal{I}} \eta_j(\mathbf{x})$ denotes the top-k targets according to the ground truth $\eta_j(\mathbf{x})$. The deviation of such an equation can be used as a suboptimality measure of $\mathcal{M}$. Formally, we define it to be the regret w.r.t. Precision@k and denote it as $\operatorname{reg}_{p@k}(\mathcal{M})$. This is a special case when $m = k$ for a more general definition $\operatorname{reg}_{p@m}(\mathcal{M}) =$

$$\mathbb{E}_{p(\mathbf{x})}\left[\frac{1}{m}\left(\sum_{j \in \mathcal{I}_{\mathbf{x}}^{(m)}} \eta_j(\mathbf{x}) - \sum_{n \in \mathcal{B}_H^{(m)}(\mathbf{x})} \eta_{\pi(n)}(\mathbf{x})\right)\right], \tag{10}$$

[7] Without any formal proof, Zhu et al. (2018) proposes the max-heap like formulation, which can be regarded as a special case of Proposition 1 with the $|\mathcal{I}_{\mathbf{x}}| = 1$ restriction. We provide a detailed proof for Proposition 1 in the supplementary materials.

where $\mathcal{I}_{\mathbf{x}}^{(m)} = \operatorname{argTopm}_{j \in \mathcal{I}} \eta_j(\mathbf{x})$.

Though $\operatorname{reg}_{p@k}(\mathcal{M})$ seems an ideal suboptimality measure, finding its minimizer is hard due to the existence of a series of nested non-differentiable argTopk operators. Therefore, finding a surrogate loss for $\operatorname{reg}_{p@k}(\mathcal{M})$ such that its minimizer is still an optimal tree model becomes very important. To distinguish such a surrogate loss, we introduce the concept of calibration under beam search as follows:

Definition 2 (Calibration under Beam Search)**.** *Given a tree model $\mathcal{M}(\mathcal{T}, g)$, a loss function $L : \{0, 1\}^M \times \mathbb{R}^{|\mathcal{N}|} \to \mathbb{R}$ is called top-k calibrated under beam search if*

$$\operatorname*{argmin}_g \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[L(\mathbf{y}, \mathbf{g}(\mathbf{x}))\right] \subset \operatorname*{argmin}_g \operatorname{reg}_{p@k}(\mathcal{M}), \tag{11}$$

holds for any distribution $p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$. L is called calibrated under beam search if Eq. (11) holds for any $1 \le k \le M$.

Definition 2 shows a tree model $\mathcal{M}(\mathcal{T}, g)$ with g minimizing a non-calibrated loss is not Bayes optimal under beam search in general. Recall that Proposition 1 shows that for any $p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ and any $\mathcal{T}$, the minimizer of $\operatorname{reg}_{p@k}(\mathcal{M})$ always exists, which satisfies $p_g(z_n|\mathbf{x}) = p^*(z_n|\mathbf{x})$ and achieves $\operatorname{reg}_{p@k}(\mathcal{M}) = 0$. Therefore, the suboptimality of TDMs and PLTs can be proved by showing the minimizer of their training loss does not guarantee $\operatorname{reg}_{p@k}(\mathcal{M}) = 0$ in general. This can be proved by finding a counterexample and the toy experiment shown in Table 1 meets this requirement. As a result, we have

Proposition 2. *Eq. (2) with z_n defined in Eq. (1) is not calibrated under beam search in general.*

4.3. Learning Optimal Tree Models under Beam Search

Given the discussion in Sec. 4.2, we need a new surrogate loss function such that its minimizer corresponds to the tree model which is Bayes optimal under beam search. According to Definition 1, when the retrieval performance is measured by Precision@m, requiring a model to be top-m Bayes optimal under beam search will be enough. Proposition 1 provides a natural surrogate loss to achieve this purpose with beam size $k \ge m$, i.e.,

$$g \in \operatorname*{argmin}_g \mathbb{E}_{p(\mathbf{x})}\left[\sum_{h=1}^{H} \sum_{n \in \tilde{\mathcal{B}}_h(\mathbf{x})} \mathrm{KL}(p^*(z_n|\mathbf{x}) \| p_g(z_n|\mathbf{x}))\right], \tag{12}$$

where we follow the TDM style and assume $p_g(z_n|\mathbf{x}) = 1/(1 + \exp(-(2z_n - 1)g(\mathbf{x}, n)))$.

Unlike Eq. (2), Eq. (12) uses nodes in $\tilde{\mathcal{B}}_h(\mathbf{x})$ instead of $\mathcal{S}_h^+(\mathbf{y})$ for training and introduces a different definition of pseudo targets compared to Eq. (1). Let $z_n^* \sim p^*(z_n|\mathbf{x})$

denote the corresponding pseudo target, we have

$$z_n^* = y_{\pi(n')},\ n' \in \operatorname*{argmax}_{n' \in \mathcal{L}(n)} \eta_{\pi(n')}(\mathbf{x}). \tag{13}$$

Notice that for $n \in \mathcal{N}_H$, $z_n^* = y_{\pi(n)}$ as well as z_n in Eq. (1). To distinguish z_n^* from z_n, we call it the optimal pseudo target since it corresponds to the optimal tree model. Given this definition, Eq. (12) can be rewritten as $\operatorname{argmin}_g \mathbb{E}_{p(\mathbf{x},\mathbf{y})} \left[L_p(\mathbf{y}, \mathbf{g}(\mathbf{x})) \right]$ where

$$L_p(\mathbf{y}, \mathbf{g}(\mathbf{x})) = \sum_{h=1}^{H} \sum_{n \in \tilde{\mathcal{B}}_h(\mathbf{x})} \ell_{\text{BCE}}(z_n^*, g(\mathbf{x}, n)). \tag{14}$$

Notice that in Eq. (14) we assign a subscript p to highlight the dependence of z_n^* on $\eta_j(\mathbf{x})$, which implies that Eq. (14) is calibrated under beam search in the sense that its formulation depends on $p : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$.

Figure 1 provides a concrete example for the difference between z_n^* and z_n. Not all ancestor nodes of a relevant target $y_j = 1$ are regarded as relevant nodes according to z_n^*: Node 1 and 6 (red nodes in Figure 1(a)) are assigned with $z_n = 1$ but with $z_n^* = 0$ (green nodes in Figure 1(c)). The reason is that among targets on the subtree rooted at these nodes, the irrelevant target has a higher $\eta_j(\mathbf{x})$ compared to the relevant target, i.e., $\eta_7(\mathbf{x}) = 0.5 > \eta_8(\mathbf{x}) = 0.4$ and $\eta_1(\mathbf{x}) = 0.8 > \eta_3(\mathbf{x}) = 0.7$, which leads z_n^* to be 0.

However, it is impossible to minimize Eq. (14) directly, since $\eta_j(\mathbf{x})$ is unknown in practice. As a result, we need to find an approximation of z_n^* without the dependence on $\eta_j(\mathbf{x})$. Suppose $g(\mathbf{x}, n)$ is parameterized with trainable parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, we use the notation $g_{\boldsymbol{\theta}}(\mathbf{x}, n)$, $p_{g_{\boldsymbol{\theta}}}(\mathbf{x})$ and $\mathcal{B}_h(\mathbf{x}; \boldsymbol{\theta})$ to highlight their dependence on $\boldsymbol{\theta}$. A natural choice is to replace $\eta_{\pi(n')}(\mathbf{x})$ in Eq. (13) with $p_{g_{\boldsymbol{\theta}}}(z_{n'} = 1|\mathbf{x})$. However, this formulation is still impractical since the computational complexity of traversing $\mathcal{L}(n)$ for each $n \in \tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta})$ is unacceptable. Thanks to the tree structure, we can approximate z_n^* with $\hat{z}_n(\mathbf{x}; \boldsymbol{\theta})$, which is constructed in a recursive manner for $n \in \mathcal{N} \setminus \mathcal{N}_H$ as

$$\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}) = \hat{z}_{n'}(\mathbf{x}; \boldsymbol{\theta}),\ n' \in \operatorname*{argmax}_{n' \in \mathcal{C}(n)} p_{g_{\boldsymbol{\theta}}}(z_{n'} = 1|\mathbf{x}), \tag{15}$$

and is set directly as $\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}) = y_{\pi(n)}$ for $n \in \mathcal{N}_H$.

By doing so, we remove the dependence on unknown $\eta_j(\mathbf{x})$. But minimizing Eq. (14) when replacing z_n^* with $\hat{z}_n(\mathbf{x}, \boldsymbol{\theta})$ is still not an easy task since the parameter $\boldsymbol{\theta}$ affects $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta})$, $\hat{z}_n(\mathbf{x}, \boldsymbol{\theta})$ and $g_{\boldsymbol{\theta}}(\mathbf{x}, n)$: Gradient with respect to $\boldsymbol{\theta}$ cannot be computed directly due to the non-differentiability of the argTopk operator in $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta})$ and the argmax operator in $\hat{z}_n(\mathbf{x}; \boldsymbol{\theta})$. To get a differentiable loss function, we propose to replace $L_p(\mathbf{y}, \mathbf{g}(\mathbf{x}))$ defined in Eq. (14) with

$$L_{\boldsymbol{\theta}_t}(\mathbf{y}, \mathbf{g}(\mathbf{x}); \boldsymbol{\theta}) = \sum_{h=1}^{H} \sum_{n \in \tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)} \ell_{\text{BCE}}(\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}_t), g_{\boldsymbol{\theta}}(\mathbf{x}, n)), \tag{16}$$

where $\boldsymbol{\theta}_t$ denotes the fixed parameter, which can be the parameter of the last iteration in a gradient based algorithm. Given the discussion above, we propose a novel algorithm for learning such a tree model as Algorithm 1.

Algorithm 1 Learning Optimal Tree Models under Beam Search

Input: Training dataset $\mathcal{D}_{tr}$, initial tree model $\mathcal{M}(\mathcal{T}, g_{\boldsymbol{\theta}_0})$ with the tree structure $\mathcal{T}$ and the node-wise scorer $g_{\boldsymbol{\theta}_0}(\mathbf{x}, n)$, beam size k, stepsize ϵ.
Output: Trained tree model $\mathcal{M}(\mathcal{T}, g_{\boldsymbol{\theta}_t})$.

1: Initialize $t = 0$;
2: **while** convergence condition is not attained **do**
3: Draw a minibatch MB from $\mathcal{D}_{tr}$;
4: Draw $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)$ according to Eq. (3);
5: Compute $\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}_t)$ for each $n \in \tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)$ according to Eq. (15);
6: Update $\boldsymbol{\theta}_{t+1}$ using a gradient based method with stepsize ϵ, e.g., ADAM (Kingma & Ba, 2015), on the current $\boldsymbol{\theta}_t$ and the gradient $\mathbf{g}_t$, where

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} \sum_{(\mathbf{x},\mathbf{y}) \sim \text{MB}} L_{\boldsymbol{\theta}_t}(\mathbf{y}, \mathbf{g}(\mathbf{x}); \boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$$

according to Eq. (16);
7: $t \leftarrow t + 1$;
8: **end while**
9: Return $\mathcal{M}(\mathcal{T}, g_{\boldsymbol{\theta}_t})$.

As is analyzed in the supplementary materials, the training complexity of Algorithm 1 is $\mathrm{O}(Hbk + Hb|\mathcal{I}_{\mathbf{x}}|)$ per instance, which is still logarithmic to M. Besides, for the tree model trained according to Algorithm 1, its testing complexity is $\mathrm{O}(Hbk)$ per instance as that in Sec. 3.2.2, since Algorithm 1 does not alter beam search in testing.

Now, the remaining question is, since introducing several approximations into Eq. (16), does it still have the nice property to achieve Bayes optimality under beam search? We provide an answer[8] as follows:

Proposition 3 (Practical Algorithm)**.** *Suppose* $\mathcal{G} = \{g_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ *has enough capacity and* $L^*_{\boldsymbol{\theta}_t}(\mathbf{y}, \mathbf{g}(\mathbf{x}); \boldsymbol{\theta}) =$

$$\sum_{h=1}^{H} \sum_{n \in \mathcal{N}_h} w_n(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}_t) \ell_{\text{BCE}}(\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}_t), g_{\boldsymbol{\theta}}(\mathbf{x}, n)), \tag{17}$$

where $w_n(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}_t) > 0$*. For any probability* $p : \mathcal{X} \times \mathcal{Y} \to$

[8] Proof can be found in the supplementary materials.

$\mathbb{R}^+$, *if there exists* $\boldsymbol{\theta}_t \in \boldsymbol{\Theta}$ *such that*

$$\boldsymbol{\theta}_t \in \operatorname*{argmin}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_{p(\mathbf{x},\mathbf{y})} \left[L^*_{\boldsymbol{\theta}_t}(\mathbf{y}, \mathbf{g}(\mathbf{x}); \boldsymbol{\theta}) \right], \tag{18}$$

the corresponding tree model $\mathcal{M}(\mathcal{T}, g_{\boldsymbol{\theta}_t})$ *is Bayes optimal under beam search.*

Proposition 3 shows that replacing z^*_n with $\hat{z}_n(\mathbf{x}, \boldsymbol{\theta})$ and introducing the fixed parameter $\boldsymbol{\theta}_t$ does not affect the optimality of $\mathcal{M}(\mathcal{T}, g_{\boldsymbol{\theta}_t})$ on Eq. (17). However, Eq. (16) does not have such a guarantee, since the summation over $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)$ corresponds to the summation over $\mathcal{N}_h$ with weight $w_n(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}_t) = \mathbb{I}(n \in \tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t))$ and thus violating the restriction that $w_n(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}_t) > 0$. This problem can be solved by introducing randomness into Eq. (16) such that each $n \in \mathcal{N}_h$ has a non-zero $w_n(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}_t)$ in expectation. Examples include adding random samples of $\mathcal{N}_h$ into the summation in Eq. (16) or leveraging stochastic beam search (Kool et al., 2019) to generate $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)$. Nevertheless, in experiments we find these strategies do not greatly affect the performance, and thus we still use Eq. (16).

5. Experiments

In this section, we experimentally verify our analysis and evaluate the performance of different tree models on both synthetic and real data. Throughout experiments, we use OTM to denote the tree model trained according to Algorithm 1 since its goal is to learn optimal tree models under beam search. To perform an ablation study, we consider two variants of OTM: OTM (-BS) differs from OTM by replacing $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)$ with $\mathcal{S}_h(\mathbf{y}) = \mathcal{S}^+_h(\mathbf{y}) \bigcup \mathcal{S}^-_h(\mathbf{y})$, and OTM (-OptEst) differs from OTM by replacing $\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}_t)$ in Eq. (13) with z_n in Eq. (1). More details of experiments can be found in the supplementary materials.

5.1. Synthetic Data

Datasets: For each instance $(\mathbf{x}, \mathbf{y})$, $\mathbf{x} \in \mathbb{R}^d$ is sampled from a d-dimensional isotropic Gaussian distribution $\mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ with zero mean and identity covariance matrix, and $\mathbf{y} \in \{0, 1\}^M$ is sampled from $p(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^M p(y_j|\mathbf{x}) = \prod_{j=1}^M 1/(1 + \exp(-(2y_j - 1)\mathbf{w}_j^\top \mathbf{x} - b))$ where the weight vector $\mathbf{w}_j \in \mathbb{R}^d$ is also sampled from $\mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$. The bias b is a predefined constant[9] to control the number of non-zero entries in $\mathbf{y}$. Corresponding training and testing datasets are denoted as $\mathcal{D}_{tr}$ and $\mathcal{D}_{te}$, respectively.

Compared Models and Metric: We compare OTM with PLT and TDM. All the tree models $\mathcal{M}(\mathcal{T}, g)$ share the same tree structure $\mathcal{T}$ and the same parameterization of the node-wise scorer g. More specifically, $\mathcal{T}$ is set to be a random binary tree over $\mathcal{I}$ and $g(\mathbf{x}, n) = \boldsymbol{\theta}_n^\top \mathbf{x} + b_n$ is parameterized as a linear scorer, where $\boldsymbol{\theta}_n \in \mathbb{R}^d$ and $b_n \in \mathbb{R}$ are trainable parameters. All models are trained on $\mathcal{D}_{tr}$ and their perfomance is measured by $\widehat{\text{reg}}_{p@m}$, which is an estimation of $\text{reg}_{p@m}(\mathcal{M})$ defined in Eq. (10) by replacing the expectation over $p(\mathbf{x})$ with the summation over $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{te}$.

[9] In experiment, we set b to be a negative value such that the number of non-zero entries is less than $0.1M$ to simulate the practical case where the number of relevant targets is much smaller than the target set size.

Table 2. A comparison of $\widehat{\text{reg}}_{p@m}(\mathcal{M})$ averaged by 5 runs with random initialization with hyperparameter settings $M = 1000$, $d = 10$, $b = -5$, $|\mathcal{D}_{tr}| = 10000$, $|\mathcal{D}_{te}| = 1000$ and $k = 50$.

m	1	10	20	50
PLT	0.0444	0.0778	0.0955	0.1492
TDM	0.0033	0.0205	0.0453	0.1363
OTM	**0.0024**	**0.0163**	**0.0349**	**0.1083**
OTM (-BS)	0.0048	0.0201	0.0421	0.1313
OTM (-OptEst)	0.0033	0.0198	0.0418	0.1218

Results: Table 2 shows that OTM performs the best compared to other models, which indicates that eliminating the training-testing discrepancy can improve retrieval performance of tree models. Both OTM (-BS) and OTM (-OptEst) have smaller regret than PLT and TDM, which means that using beam search aware subsampling (i.e., $\tilde{\mathcal{B}}_h(\mathbf{x}; \boldsymbol{\theta}_t)$) or estimated optimal pseudo targets (i.e., $\hat{z}_n(\mathbf{x}; \boldsymbol{\theta}_t)$) alone contributes to better performance. Besides, OTM (-OptEst) has smaller regret than OTM (-BS), which reveals that beam search aware subsampling contributes more than estimated optimal pseudo targets to the performance of OTM.

5.2. Real Data

Datasets: Our experiment are conducted on two large-scale real datasets for recommendation tasks: Amazon Books (McAuley et al., 2015; He & McAuley, 2016) and UserBehavior (Zhu et al., 2018). Each record of both datasets is organized in the format of user-item interaction, which contains user ID, item ID and timestamp. The original interaction records are formulated as a set of user-based data. Each user-based data is denoted as a list of items sorted by the timestep that the user-item interaction occurs. We discard the user based data which has less than 10 items and split the rest into training set $\mathcal{D}_{tr}$, validation set $\mathcal{D}_{val}$ and testing set $\mathcal{D}_{te}$ in the same way as Zhu et al. (2018; 2019). For the validation and testing set, we take the first half of each user-based data according to ascending order along timestamp as the feature $\mathbf{x}$, and the latter half as the relevant targets $\mathbf{y}$. While training instances are generated from the raw user-based data considering the characteristics of different approaches on the training set. If the approach restricts $|\mathcal{I}_\mathbf{x}| = 1$, we use a sliding window to produce several instances for each user based data, while one instance

Table 3. Precision@m, Recall@m and F-Measure@m comparison on Amazon Books with beam size $k = 400$ and various m (%).

Method	Precision				Recall				F-Measure			
	10	50	100	200	10	50	100	200	10	50	100	200
Item-CF	2.02	1.04	0.74	0.52	2.14	4.71	6.29	8.18	1.92	1.55	1.23	0.92
YouTube product-DNN	1.26	0.84	0.67	0.53	1.12	3.52	5.41	8.26	1.05	1.21	1.09	0.93
HSM	1.50	0.93	0.73	0.54	1.25	3.59	5.59	8.04	1.21	1.30	1.18	0.95
PLT	1.85	1.26	0.99	0.75	1.57	4.87	7.35	10.59	1.48	1.74	1.57	1.29
JTM	1.84	1.34	1.07	0.80	1.75	5.79	8.70	12.60	1.60	1.94	1.73	1.40
OTM	**3.12**	**1.97**	**1.49**	**1.06**	**2.76**	**8.16**	**11.86**	**16.36**	**2.58**	**2.80**	**2.39**	**1.86**
OTM (-BS)	2.18	1.45	1.15	0.86	1.91	6.01	9.40	13.68	1.81	2.08	1.88	1.52
OTM (-OptEst)	3.07	1.92	1.45	1.05	2.70	8.00	11.63	16.17	2.54	2.74	2.33	1.83

Table 4. Precision@m, Recall@m and F-Measure@m comparison on UserBehavior with beam size $k = 400$ and various m (%).

Method	Precision				Recall				F-Measure			
	10	50	100	200	10	50	100	200	10	50	100	200
Item-CF	5.45	3.07	2.20	1.56	1.25	3.31	4.74	6.75	1.84	2.76	2.64	2.30
YouTube product-DNN	9.04	4.52	3.22	2.25	2.29	5.36	7.49	10.15	3.29	4.23	3.97	3.36
HSM	9.79	4.49	3.04	2.01	2.58	5.60	7.38	9.52	3.68	4.30	3.80	3.03
PLT	11.47	5.07	3.47	2.35	2.85	5.84	7.75	10.22	4.13	4.72	4.23	3.48
JTM	20.05	7.45	4.85	3.12	5.42	9.39	11.84	14.75	7.62	7.15	6.06	4.70
OTM	**22.47**	**8.21**	**5.33**	**3.42**	**5.95**	**10.07**	**12.62**	**15.68**	**8.40**	**7.78**	**6.59**	**5.12**
OTM (-BS)	19.81	7.74	5.08	3.31	5.36	9.57	12.14	15.29	7.54	7.36	6.30	4.95
OTM (-OptEst)	22.38	8.20	**5.33**	3.40	5.92	10.06	12.61	15.61	8.36	**7.78**	**6.59**	5.08

is obtained for methods without restriction on $|\mathcal{I}_{\mathbf{x}}|$.

Compared Models and Metric: We compare OTM with two series of methods: (1) widely used methods in recommendation tasks, such as Item-CF (Sarwar et al., 2001), the basic collaborative filtering method, and YouTube product-DNN (Covington et al., 2016), the representative work of vector kNN based methods; (2) tree models like HSM (Morin & Bengio, 2005), PLT and JTM (Zhu et al., 2019). HSM is a hierarchical softmax model which can be regarded as PLT with the $|\mathcal{I}_{\mathbf{x}}| = 1$ restriction. JTM is a variant of TDM which trains tree structure and node-wise scorers jointly and achieves state-of-the-art performance on these two datasets. All the tree models share the same binary tree structure and adopt the same neural network model for node-wise scorers. The neural network consists of three fully connected layers with hidden size 128, 64 and 24 and parametric ReLU is used as the activation function. The performance of different models is measured by Precision@m (Eq. (5)), Recall@m (Eq. (6)) and F-Measure@m (Eq. (7)) averaged over the testing set $\mathcal{D}_{te}$.

Results: Table 3 and Table 4 show results of Amazon Books and UserBehavior, respectively[10]. Our model performs the best among all methods: Compared to the previous state-of-the-art JTM, OTM achieves 29.8% and 6.3% relative recall lift ($m = 200$) on Amazon Books and UserBehavior separately. Results of OTM and its two variants are consistent with that on synthetic data: Both beam search aware subsampling and estimated optimal pseudo targets contribute to better performance, while the former contributes more and the performance of OTM mainly depends on the former. Besides, the comparison between HSM and PLT also demonstrates that removing the restriction of $|\mathcal{I}_{\mathbf{x}}| = 1$ in tree models contributes to performance improvement.

To understand why OTM achieves more significant improvement (29.8% versus 6.3%) on Amazon Books than UserBehavior, we analyze the statistics of these datasets and their corresponding tree structure. For each $n \in \mathcal{N}$, we define $S_n = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_{tr}} z_n$ to count the number of training instances which are relevant to n (i.e., $z_n = 1$). For each level $1 \le h \le H$, we sort $\{S_n : n \in \mathcal{N}_h\}$ in a descending order and normalize them as $S_n / \sum_{n'\in\mathcal{N}_h} S_{n'}$. This produces a level-wise distribution, which reflects the data imbalance on relevant nodes resulted from the intrinsic property of both the datasets and the tree structure. As is shown in Figure 2, the level-wise distribution of UserBehavior has a

[10] As $k = 400$, OTM is trained on $|\tilde{\mathcal{B}}_h(\mathbf{x};\boldsymbol{\theta}_t)| = 800$ nodes per level. For fairness in comparison, JTM also subsample $|\mathcal{S}_h(\mathbf{y})| =$ 800 nodes per level for training $g(\mathbf{x}, n)$.

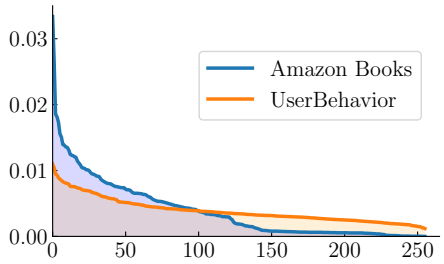

Figure 2. Results of $S_n / \sum_{n' \in \mathcal{N}_h} S_{n'}$ versus sorted node index on Amazon Books and UserBehavior with $h = 8$ ($|\mathcal{N}_h| = 256$).

heavier tail than that of Amazon Books at the same level. This implies the latter has a higher proportion of instances concentrated on only parts of nodes, which makes it easier for beam search to retrieve relevant nodes for training and thus leads to more significant improvement.

To verify our analysis on the time complexity of tree models, we compare their empirical training time, since they share the same beam search process in testing. More specifically, we compute the wall-clock time per batch for training PLT, TDM and PLT with batch size 100 on the UserBehavior dataset. This number is averaged over 5000 training iterations on a single Tesla P100-PCIE-16GB GPU. The results are 0.184s for PLT, 0.332s for TDM and 0.671s for OTM, respectively. Though OTM costs longer time than PLT and JTM, they have the same order of magnitude. This is not weird, since the step 4 and 5 in Algorithm 1 only increases the constant factor of complexity. Besides, this is a reasonable trade-off for better performance and distributed training can alleviate this in practical applications.

6. Conclusions and Future Work

Tree models have been widely adopted in large-scale information retrieval and recommendation tasks due to their logarithmic computational complexity. However, little attention has been paid to the training-testing discrepancy where the retrieval performance deterioration caused by beam search in testing is ignored in training. To the best of our knowledge, we are the first to study this problem on tree models theoretically. We also propose a novel training algorithm for learning optimal tree models under beam search which achieves improved experiment results compared to the state-of-the-arts on both synthetic and real data.

For future work, we'd like to explore other techniques for training $g(\mathbf{x}, n)$ according to Eq. (14), e.g., the REINFORCE algorithm (Williams, 1992; Ranzato et al., 2016) and the actor-critic algorithm (Sutton et al., 2000; Bahdanau et al., 2017). We also want to extend our algorithm for learning tree structure and node-wise scorers jointly. Besides, applying our algorithm to applications like extreme multilabel text classification is also an interesting direction.

Acknowledgements

We deeply appreciate Xiang Li, Rihan Chen, Daqing Chang, Pengye Zhang, Jie He and Xiaoqiang Zhu for their insightful suggestions and discussions. We thank Huimin Yi, Yang Zheng, Siran Yang, Guowang Zhang, Shuai Li, Yue Song and Di Zhang for implementing the key components of the training platform. We thank Linhao Wang, Yin Yang, Liming Duan and Guan Wang for necessary supports about online serving. We thank anonymous reviewers for their constructive feedback and helpful comments.

References

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*, 2017.

Cohen, E. and Beck, C. Empirical analysis of beam search performance degradation in neural sequence models. In *International Conference on Machine Learning*, pp. 1290–1299, 2019.

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

Daumé III, H. and Marcu, D. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine learning*, pp. 169–176. ACM, 2005.

Goyal, K., Neubig, G., Dyer, C., and Berg-Kirkpatrick, T. A continuous relaxation of beam search for end-to-end training of neural sequence models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

He, R. and McAuley, J. 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*, pp. 507–517, 2016.

Jain, H., Prabhu, Y., and Varma, M. 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*, pp. 935–944, 2016.

Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., and Hullermeier, E. Extreme f-measure maximization using sparse probability estimates. In *International Conference on Machine Learning*, pp. 1435–1444, 2016.

Khandagale, S., Xiao, H., and Babbar, R. Bonsai-diverse and shallow trees for extreme multi-label classification. *arXiv preprint arXiv:1904.08249*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kool, W., Van Hoof, H., and Welling, M. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pp. 3499–3508, 2019.

Lapin, M., Hein, M., and Schiele, B. Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1533–1554, 2017.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52, 2015.

Menon, A. K., Rawat, A. S., Reddi, S., and Kumar, S. Multilabel reductions: what is my loss optimising? In *Advances in Neural Information Processing Systems*, pp. 10599–10610, 2019.

Morin, F. and Bengio, Y. Hierarchical probabilistic neural network language model. In *Proceedings of the eighth international conference on artificial intelligence and statistics*, volume 5, pp. 246–252. Citeseer, 2005.

Negrinho, R., Gormley, M., and Gordon, G. J. Learning beam search policies via imitation learning. In *Advances in Neural Information Processing Systems*, pp. 10652–10661, 2018.

Prabhu, Y. and Varma, M. 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*, pp. 263–272, 2014.

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., and Varma, M. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, pp. 993–1002. International World Wide Web Conferences Steering Committee, 2018.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wiseman, S. and Rush, A. M. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1296–1306, 2016.

Wu, X.-Z. and Zhou, Z.-H. A unified view of multi-label performance measures. In *International Conference on Machine Learning*, pp. 3780–3788. JMLR. org, 2017.

Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., and Dembczynski, K. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pp. 6355–6366, 2018.

Xu, Y. and Fern, A. On learning linear ranking functions for beam search. In *International Conference on Machine learning*, pp. 1047–1054, 2007.

Yang, F. and Koyejo, S. On the consistency of top-k surrogate losses. *arXiv preprint arXiv:1901.11141*, 2019.

You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H., and Zhu, S. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *Advances in Neural Information Processing Systems*, pp. 5812–5822, 2019.

Zhu, H., Li, X., Zhang, P., Li, G., He, J., Li, H., and Gai, K. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1079–1088. ACM, 2018.

Zhu, H., Chang, D., Xu, Z., Zhang, P., Li, X., He, J., Li, H., Xu, J., and Gai, K. Joint optimization of tree-based index and deep model for recommender systems. In *Advances in Neural Information Processing Systems*, pp. 3973–3982, 2019.