Propensity-scored Probabilistic Label Trees

Marek Wydmuch
Poznan University of Technology
Poznan, Poland
mwydmuch@cs.put.poznan.pl

Kalina Jasinska-Kobus*
ML Research at Allegro.pl
Poznan, Poland
kjasinska@cs.put.poznan.pl

Rohit Babbar
Aalto University
Helsinki, Finland
rohit.babbar@aalto.fi

Krzysztof Dembczyński*
Yahoo! Research
New York, USA
kdembczynski@cs.put.poznan.pl

ABSTRACT

Extreme multi-label classification (XMLC) refers to the task of tagging instances with small subsets of relevant labels coming from an extremely large set of all possible labels. Recently, XMLC has been widely applied to diverse web applications such as automatic content labeling, online advertising, or recommendation systems. In such environments, label distribution is often highly imbalanced, consisting mostly of very rare tail labels, and relevant labels can be missing. As a remedy to these problems, the propensity model has been introduced and applied within several XMLC algorithms. In this work, we focus on the problem of optimal predictions under this model for probabilistic label trees, a popular approach for XMLC problems. We introduce an inference procedure, based on the A^*-search algorithm, that efficiently finds the optimal solution, assuming that all probabilities and propensities are known. We demonstrate the attractiveness of this approach in a wide empirical study on popular XMLC benchmark datasets.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**.

KEYWORDS

extreme classification, multi-label classification, propensity model, missing labels, probabilistic label trees, supervised learning, recommendation, tagging, ranking

ACM Reference Format:
Marek Wydmuch, Kalina Jasinska-Kobus, Rohit Babbar, and Krzysztof Dembczyński. 2021. Propensity-scored Probabilistic Label Trees. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3404835.3463084

*Also with Poznan University of Technology.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

1 INTRODUCTION

Extreme multi-label classification (XMLC) is a supervised learning problem, where only a few labels from an enormous label space, reaching orders of millions, are relevant per data point. Notable examples of problems where XMLC framework can be effectively leveraged are tagging of text documents [8], content annotation for multimedia search [9], and diverse types of recommendation, including webpages-to-ads [5], ads-to-bid-words [2, 19], users-to-items [23, 28], queries-to-items [17], or items-to-queries [7]. These practical applications impose new statistical challenges, including: 1) long-tail distribution of labels—infrequent (tail) labels are much harder to predict than frequent (head) labels due to the data imbalance problem; 2) missing relevant labels in learning data—since it is nearly impossible to check the whole set of labels when it is so large, and the chance for a label to be missing is higher for tail than for head labels [11].

Many XMLC models achieve good predictive performance by just focusing on head labels [22]. However, this is not desirable in many of the mentioned applications (e.g., recommendation and content annotation), where tail labels might be more informative. To address this issue Jain et al. [11] proposed to evaluate XMLC models in terms of propensity-scored versions of popular measures (i.e., precision@k, recall@k, and nDCG@k). Under the propensity model, we assume that an assignment of a label to an example is always correct, but the supervision may skip some positive labels and leave them not assigned to the example with some probability (different for each label).

In this work, we introduce the Bayes optimal inference procedure for propensity-scored precision@k for probabilistic classifiers trained on observed data. While this approach can be easily applied to many classical models, we particularly show how to implement it for probabilistic label trees (PLTs) [12], an efficient and competitive approach to XMLC, being the core of many existing state-of-the-art algorithms (e.g., Parabel [18], extremeText [24], Bonsai [15], AttentionXML [25], napkinXC [13], and PECOS that includes XR-Linear [26] and X-Transformers [7] methods). We demonstrate that this approach achieves very competitive results in terms of statistical performance and running times.

2 PROBLEM STATEMENT

In this section, we state the problem. We first define extreme multi-label classification (XMLC) and then the propensity model.

2.1 Extreme multi-label classification

Let $\mathcal{X}$ denote an instance space, and let $\mathcal{L} = [m]$ be a finite set of m class labels. We assume that an instance $\boldsymbol{x} \in \mathcal{X}$ is associated with a subset of labels $\mathcal{L}_{\boldsymbol{x}} \subseteq \mathcal{L}$ (the subset can be empty); this subset is often called the set of *relevant* or *positive* labels, while the complement $\mathcal{L} \setminus \mathcal{L}_{\boldsymbol{x}}$ is considered as *irrelevant* or *negative* for $\boldsymbol{x}$. We identify the set $\mathcal{L}_{\boldsymbol{x}}$ of relevant labels with the binary vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$, in which $y_j = 1 \Leftrightarrow j \in \mathcal{L}_{\boldsymbol{x}}$. By $\mathcal{Y} = \{0,1\}^m$ we denote the set of all possible label vectors. In the classical setting, we assume that observations $(\boldsymbol{x}, \boldsymbol{y})$ are generated independently and identically according to a probability distribution $\mathbf{P}(\boldsymbol{x}, \boldsymbol{y})$ defined on $\mathcal{X} \times \mathcal{Y}$. Notice that the above definition concerns not only multi-label classification, but also multi-class (when $\|\boldsymbol{y}\|_1 = 1$) and k-sparse multi-label (when $\|\boldsymbol{y}\|_1 \leq k$) problems as special cases. In case of XMLC we assume m to be a large number (e.g., $\geq 10^5$), and $\|\boldsymbol{y}\|_1$ to be much smaller than m, $\|\boldsymbol{y}\|_1 \ll m$.[1]

The problem of XMLC can be defined as finding a *classifier* $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))$, from a function class $\mathcal{H}^m : \mathcal{X} \to \mathbb{R}^m$, that minimizes the *expected loss* or *risk*:

$$L_\ell(\boldsymbol{h}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbf{P}(\boldsymbol{x},\boldsymbol{y})}\left(\ell(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x}))\right), \tag{1}$$

where $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$ is the (*task*) *loss*. The optimal classifier, the so-called *Bayes classifier*, for a given loss function ℓ is: $\boldsymbol{h}^*_\ell = \arg\min_{\boldsymbol{h}} L_\ell(\boldsymbol{h})$.

2.2 Propensity model

In the case of XMLC, the real-world data may not follow the classical setting, which assumes that $(\boldsymbol{x}, \boldsymbol{y})$ are generated according to $\mathbf{P}(\boldsymbol{x}, \boldsymbol{y})$. As correct labeling (without any mistakes or noise) in case of an extremely large label set is almost impossible, it is reasonable to assume that positive labels can be missing [11]. Mathematically, the model can be defined in the following way. Let $\boldsymbol{y}$ be the original label vector associated with $\boldsymbol{x}$. We observe, however, $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \ldots, \tilde{y}_m)$ such that:

$$\begin{aligned} &\mathbf{P}(\tilde{y}_j = 1 \mid y_j = 1) = p_j\,, \quad &&\mathbf{P}(\tilde{y}_j = 0 \mid y_j = 1) = 1 - p_j\,, \\ &\mathbf{P}(\tilde{y}_j = 1 \mid y_j = 0) = 0\,, \quad &&\mathbf{P}(\tilde{y}_j = 0 \mid y_j = 0) = 1\,, \end{aligned} \tag{2}$$

where $p_j \in [0,1]$ is the propensity of seeing a positive label when it is indeed positive. All observations in both training and test sets do follow the above model. The propensity does not depend on $\boldsymbol{x}$. This means that for the observed conditional probability of label j, we have:

$$\tilde{\eta}_j(\boldsymbol{x}) = \mathbf{P}(\tilde{y}_j = 1 \mid \boldsymbol{x}) = p_j \mathbf{P}(y_j = 1 \mid \boldsymbol{x}) = p_j \eta_j(\boldsymbol{x})\,. \tag{3}$$

Let us denote the inverse propensity by q_j, i.e. $q_j = \frac{1}{p_j}$. Thus, the original conditional probability of label j is given by:

$$\eta_j(\boldsymbol{x}) = \mathbf{P}(y_j = 1 \mid \boldsymbol{x}) = q_j \mathbf{P}(\tilde{y}_j = 1 \mid \boldsymbol{x}) = q_j \tilde{\eta}_j(\boldsymbol{x})\,. \tag{4}$$

Therefore, we can appropriately adjust inference procedures of algorithms estimating $\tilde{\eta}_j(\boldsymbol{x})$ to act optimally under different propensity-scored loss functions.

[1] We use $[n]$ to denote the set of integers from 1 to n, and $\|\boldsymbol{x}\|_1$ to denote the L_1 norm of $\boldsymbol{x}$.

3 BAYES OPTIMAL DECISIONS FOR PROPENSITY-SCORED PRECISION@K

Jain et al. [11] introduced propensity-scored variants of popular XMLC measures. For precision@k it takes the form:

$$psp@k(\tilde{\boldsymbol{y}}, \boldsymbol{h}_{@k}(\boldsymbol{x})) = \frac{1}{k} \sum_{j \in \hat{\mathcal{L}}_{\boldsymbol{x}}} q_j [\![\tilde{y}_j = 1]\!]\,, \tag{5}$$

where $\hat{\mathcal{L}}_{\boldsymbol{x}}$ is a set of k labels predicted by $\boldsymbol{h}_{@k}$ for $\boldsymbol{x}$. Notice that precision@k ($p@k$) is a special case of $psp@k$ if $q_j = 1$ for all j.

We define a loss function for propensity-scored precision@k as $\ell_{psp@k} = -psp@k$. The conditional risk for $\ell_{psp@k}$ is then:

$$\begin{aligned} L_{psp@k}(\boldsymbol{h}_{@k} \mid \boldsymbol{x}) &= \mathbb{E}_{\tilde{\boldsymbol{y}}} \ell_{psp@k}(\tilde{\boldsymbol{y}}, \boldsymbol{h}_{@k}(\boldsymbol{x})) \\ &= -\sum_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} \mathbf{P}(\tilde{\boldsymbol{y}} \mid \boldsymbol{x}) \frac{1}{k} \sum_{j \in \hat{\mathcal{L}}_{\boldsymbol{x}}} q_j [\![\tilde{y}_j = 1]\!] \\ &= -\frac{1}{k} \sum_{j \in \hat{\mathcal{L}}_{\boldsymbol{x}}} q_j \sum_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} \mathbf{P}(\tilde{\boldsymbol{y}} \mid \boldsymbol{x}) [\![\tilde{y}_j = 1]\!] \\ &= -\frac{1}{k} \sum_{j \in \hat{\mathcal{L}}_{\boldsymbol{x}}} q_j \tilde{\eta}_j(\boldsymbol{x})\,. \end{aligned}$$

The above result shows that the Bayes optimal classifier for $psp@k$ is determined by the conditional probabilities of labels scaled by the inverse of the label propensity. Given that the propensities or their estimates are given in the time of prediction, $psp@k$ is optimized by selecting k labels with the highest values of $q_j \tilde{\eta}_j(\boldsymbol{x})$.

4 PROPENSITY-SCORED PROBABILISTIC LABEL TRESS

Conditional probabilities of labels can be estimated using many types of multi-label classifiers, such as decision trees, k-nearest neighbors, or binary relevance (BR) trained with proper composite surrogate losses, e.g., squared error, squared hinge, logistic or exponential loss [1, 27]. For such models, where estimates of $\tilde{\eta}_j(\boldsymbol{x})$ are available for all $j \in \mathcal{L}$, application of the Bayes decision rule for propensity-scored measures is straightforward. However, in many XMLC applications, calculating the full set of conditional probabilities is not feasible. In this section, we introduce an algorithmic solution of applying the Bayes decision rule for $psp@k$ to probabilistic label trees (PLTs).

4.1 Probabilistic labels trees (PLTs)

We denote a tree by T, a set of all its nodes by V_T, a root node by r_T, and the set of leaves by L_T. The leaf $l_j \in L_T$ corresponds to the label $j \in \mathcal{L}$. The parent node of v is denoted by $\mathrm{pa}(v)$, and the set of child nodes by $\mathrm{Ch}(v)$. The set of leaves of a (sub)tree rooted in node v is denoted by L_v, and path from node v to the root by $\mathrm{Path}(v)$.

A PLT uses tree T to factorize conditional probabilities of labels, $\eta_j(x) = \mathbf{P}(y_j = 1|\boldsymbol{x})$, $j \in \mathcal{L}$, by using the chain rule. Let us define an event that $\mathcal{L}_{\boldsymbol{x}}$ contains at least one relevant label in L_v: $z_v = (|\{j : l_j \in L_v\} \cap \mathcal{L}_{\boldsymbol{x}}| > 0)$. Now for every node $v \in V_T$, the conditional probability of containing at least one relevant label is given by:

$$\mathbf{P}(z_v = 1|\boldsymbol{x}) = \eta_v(\boldsymbol{x}) = \prod_{v' \in \mathrm{Path}(v)} \eta(\boldsymbol{x}, v')\,, \tag{6}$$

where $\eta(\boldsymbol{x}, v) = \mathbf{P}(z_v = 1 | z_{\mathrm{pa}(v)} = 1, \boldsymbol{x})$ for non-root nodes, and $\eta(\boldsymbol{x}, v) = \mathbf{P}(z_v = 1 \mid \boldsymbol{x})$ for the root. Notice that (6) can also be stated as recursion:

$$\eta_v(\boldsymbol{x}) = \eta(\boldsymbol{x}, v)\eta_{\mathrm{pa}(v)}(\boldsymbol{x}), \tag{7}$$

and that for leaf nodes we get the conditional probabilities of labels:

$$\eta_{l_j}(\boldsymbol{x}) = \eta_j(\boldsymbol{x}), \quad \text{for } l_j \in L_T. \tag{8}$$

To obtain a PLT, it suffices for a given T to train probabilistic classifiers from $\mathcal{H} : \mathbb{R}^d \mapsto [0, 1]$, estimating $\eta(\boldsymbol{x}, v)$ for all $v \in V_T$. We denote estimates of η by $\hat{\eta}$. We index this set of classifiers by the elements of V_T as $H = \{\hat{\eta}(v) \in \mathcal{H} : v \in V_T\}$.

4.2 Plug-in Bayes optimal prediction PLTs

An inference procedure for PLTs, based on UNIFORM-COST SEARCH, has been introduced in [12]. It efficiently finds k leaves, with highest $\hat{\eta}_j(\boldsymbol{x})$ values. Since inverse propensity is larger than one, the same method cannot be reliably applied to find leaves with the k highest products of q_j and $\hat{\tilde{\eta}}_j(\boldsymbol{x})$. To do it, we modify this procedure to an A^*-SEARCH-style algorithm. To this end we introduce cost function $f(l_j, \boldsymbol{x})$ for each path from the root to a leaf. Notice that:

$$q_j \hat{\tilde{\eta}}_j(\boldsymbol{x}) = \exp\left(- \left(- \log q_j - \sum_{v \in \mathrm{Path}(l_j)} \log \hat{\tilde{\eta}}(\boldsymbol{x}, v) \right) \right). \tag{9}$$

This allows us to use the following definition of the cost function:

$$f(l_j, \boldsymbol{x}) = \log q_{\max} - \log q_j - \sum_{v \in \mathrm{Path}(l_j)} \log \hat{\tilde{\eta}}(\boldsymbol{x}, v), \tag{10}$$

where $q_{\max} = \max_{j \in \mathcal{L}} q_j$ is a natural upper bound of $q_j \hat{\tilde{\eta}}_j(\boldsymbol{x})$ for all paths. We can then guide the A*-SEARCH with function $\hat{f}(v, \boldsymbol{x}) = g(v, \boldsymbol{x}) + h(v, \boldsymbol{x})$, estimating the value of the optimal path, where:

$$g(v, \boldsymbol{x}) = - \sum_{v' \in \mathrm{Path}(v)} \log \hat{\tilde{\eta}}(\boldsymbol{x}, v') \tag{11}$$

is a cost of reaching tree node v from the root, and:

$$h(v, \boldsymbol{x}) = \log q_{\max} - \log \max_{j \in \mathcal{L}_v} q_j \tag{12}$$

is a heuristic function estimating the cost of reaching the best leaf node from node v. To guarantee that A^*-SEARCH finds the optimal solution—top-k labels with the highest $f(l_j, \boldsymbol{x})$ and thereby top-k labels with the highest $q_j \hat{\tilde{\eta}}_j(\boldsymbol{x})$—we need to ensure that $h(v, \boldsymbol{x})$ is admissible, i.e., it never overestimates the cost of reaching a leaf node [21]. We also would like $h(v, \boldsymbol{x})$ to be consistent, making the A^*-SEARCH optimally efficient, i.e., there is no other algorithm used with the heuristic that expands fewer nodes [21]. Notice that the heuristic function assumes that probabilities estimated in nodes in a subtree rooted in v are equal to 1. Since $\log 1 = 0$, the heuristic comes to finding the label in the subtree of v with the largest value of the inverse propensity.

Algorithm 1 outlines the prediction procedure for PLTs that returns the top-k labels with the highest values of $q_j \hat{\tilde{\eta}}_j(\boldsymbol{x})$. We call this algorithm Propensity-scored PLTs (PS-PLTs). The algorithm is very similar to the original UNIFORM-COST SEARCH prediction procedure used in PLTs, which finds the top-k labels with the highest $\hat{\eta}_j(\boldsymbol{x})$. The difference is that nodes in PS-PLT are evaluated in the ascending order of their estimated cost values $\hat{f}(v, \boldsymbol{x})$ instead of decreasing conditional probabilities $\hat{\eta}_v(\boldsymbol{x})$.

Theorem 1. For any $T, H, \boldsymbol{q}$, and $\boldsymbol{x}$ the Algorithm 1 is admissible and optimally efficient.

PROOF. A^*-SEARCH finds an optimal solution if the heuristic h is admissible, i.e., if it never overestimates the true value of h^*, the cost value of reaching the best leaf in a subtree of node v. For node $v \in V$, we have:

$$h^*(v, \boldsymbol{x}) = \log q_{\max} - \log \max_{j \in \mathcal{L}_v} q_j - \sum_{v' \in \mathrm{Path}(l_j) \setminus \mathrm{Path}(v)} \log \hat{\tilde{\eta}}(\boldsymbol{x}, v'). \tag{13}$$

Since $\hat{\tilde{\eta}}(\boldsymbol{x}, v) \in [0, 1]$ and therefore $\log \hat{\tilde{\eta}}(\boldsymbol{x}, v) \leq 0$, we have that $h^*(v, \boldsymbol{x}) \geq h(v, \boldsymbol{x})$, for all $v \in V_T$, which proves admissibility.

A^*-SEARCH is optimally efficient if $h(v, \boldsymbol{x})$ is consistent (monotone), i.e., its estimate is always less than or equal to the estimate for any child node plus the cost of reaching that child. Since we have that $\max_{j \in L_{\mathrm{pa}(v)}} q_j \geq \max_{j \in L_v} q_j$, and the cost of reaching v from $\mathrm{pa}(v)$ is $-\log(\hat{\tilde{\eta}}(\boldsymbol{x}, v))$ which is greater or equal 0, it holds that $h(\mathrm{pa}(v), \boldsymbol{x}) \leq h(v, \boldsymbol{x}) - \log(\hat{\tilde{\eta}}(\boldsymbol{x}, v))$. □

The same cost function $f(l_j, \boldsymbol{x})$ can be used with other tree inference algorithms (for example discussed by Jasinska-Kobus et al. [13]), including BEAM SEARCH [16], that is approximate method for finding k leaves with highest $\hat{\eta}_j(\boldsymbol{x})$. It is used in many existing label tree implementations such as PARABEL, BONSAI, ATTENTIONXML and PECOS. We present BEAM SEARCH variant of PS-PLT in the Appendix.

5 EXPERIMENTAL RESULTS

In this section, we empirically show the usefulness of the proposed plug-in approach by incorporating it into BR and PLT algorithms and comparing these algorithms to their vanilla versions and state-of-the-art methods, particularly those that focus on tail-labels performance: PFASTREXML [11], PROXML [4], a variant of DISMEC [3] with a re-balanced and unbiased loss function as implemented in PW-DISMEC [20] (class-balanced variant), and PARABEL [18]. We conduct a comparison on six well-established XMLC benchmark datasets from the XMLC repository [6], for which we use the original train and test splits. Statistics of the used datasets can be found in the Appendix. For algorithms listed above, we report results as found in respective papers.

Since true propensities are unknown for the benchmark datasets, as true $\boldsymbol{y}$ is unavailable due to the large label space, for empirical evaluation we model propensities as proposed by Jain et al. [11]:

$$p_j = \mathbf{P}(\tilde{y}_j = 1 \mid y_j = 1) = \frac{1}{q_j} = \frac{1}{1 + Ce^{-A \log(N_j + B)}}, \tag{14}$$

where N_j is the number of data points annotated with label j in the observed ground truth dataset of size N, parameters A and B are specific for each dataset, and $C = (\log N - 1)(B + 1)^A$. We calculate propensity values on train set for each dataset using parameter values recommended in [11]. Values of A and B are included in Table 1. We evaluate all algorithms with both propensity-scored and standard precision@k.

Algorithm 1 PS-PLT.PredictTopLabels($T, H, \boldsymbol{q}, \boldsymbol{x}, k$)

$\hat{\boldsymbol{y}} = \boldsymbol{0}$, $q_{\max} = \max_{j \in \mathcal{L}} q_j$, $Q = \emptyset$, ▷ Initialize prediction $\hat{\boldsymbol{y}}$ vector to all zeros, $q_{\max}$ and a priority queue Q, ordered ascending by $\hat{f}(v, \boldsymbol{x})$
$g(r_T, \boldsymbol{x}) = -\log \hat{\eta}(\boldsymbol{x}, r_T)$ ▷ Calculate cost $g(r_T, \boldsymbol{x})$ for the tree
$\hat{f}(r_T, \boldsymbol{x}) = g(r_T, \boldsymbol{x}) + \log q_{\max} - \log \max_{j \in \mathcal{L}_{r_T}} q_j$ ▷ Calculate estimated cost $\hat{f}(r_T, \boldsymbol{x})$ for the tree root
Q.add($(r_T, g(r_T, \boldsymbol{x}), \hat{f}(r_T, \boldsymbol{x}))$) ▷ Add the tree root with cost $g(r_T, \boldsymbol{x})$ and estimation $\hat{f}(r_T, \boldsymbol{x})$ to the queue
while $\|\hat{\boldsymbol{y}}\|_1 < k$ **do** ▷ While the number of predicted labels is less than k
  $(v, g(v, \boldsymbol{x}), \_) = Q$.pop() ▷ Pop the element with the lowest cost from the queue (only node and corresponding probability)
  if v is a leaf **then** $\hat{y}_v = 1$ ▷ If the node is a leaf, set the corresponding label in the prediction vector
  else for $v' \in \text{Ch}(v)$ **do** ▷ If the node is an internal node, for all child nodes
    $g(v', \boldsymbol{x}) = g(v, \boldsymbol{x}) - \log \hat{\eta}(\boldsymbol{x}, v')$ ▷ Compute $g(v', \boldsymbol{x})$ using $\hat{\eta}(v', \boldsymbol{x}) \in H$
    $\hat{f}(v', \boldsymbol{x}) = g(v', \boldsymbol{x}) + \log q_{\max} - \log \max_{j \in \mathcal{L}_{v'}} q_j$ ▷ Calculate estimation $\hat{f}(v', \boldsymbol{x})$
    Q.add($(v', g(v', \boldsymbol{x}), \hat{f}(v', \boldsymbol{x}))$) ▷ Add the node, computed cost $g(v', \boldsymbol{x})$, and estimation $\hat{f}(v', \boldsymbol{x})$ to the queue
return $\hat{\boldsymbol{y}}$ ▷ Return the prediction vector

Table 1: PS-PLTs and PLTs compared to other state-of-the-art algorithms on propensity-scored and standard precision@{1, 3, 5} [%]. The best result for each measure is in bold. The best result in the group of sub-linear methods (the last 4 methods) is underlined.

Algorithm	*psp*@1	*psp*@3	*psp*@5	*p*@1	*p*@3	*p*@5	*psp*@1	*psp*@3	*psp*@5	*p*@1	*p*@3	*p*@5	*psp*@1	*psp*@3	*psp*@5	*p*@1	*p*@3	*p*@5
	EurLex-4K, $A = 0.55, B = 1.5$						AmazonCat-13K, $A = 0.55, B = 1.5$						Wiki10-31K, $A = 0.55, B = 1.5$					
ProXML	45.20	48.50	51.00	**86.50**	68.40	53.20	results not reported						results not reported					
PW-DiSMEC	43.48	48.81	51.25	82.25	68.80	57.18	64.95	71.35	74.37	**93.54**	78.50	63.33	12.67	15.87	18.28	**85.77**	**78.17**	**68.53**
BR	36.67	44.54	49.05	81.91	**68.85**	**57.83**	51.54	64.16	71.20	92.89	78.35	63.69	12.03	13.24	14.07	84.49	72.50	63.23
PS-BR	**46.13**	**49.60**	**51.78**	78.45	68.01	57.62	66.00	71.28	74.08	86.55	76.22	63.15	19.24	17.69	17.60	80.61	69.70	61.86
PfastreXML	43.86	45.72	46.97	75.45	62.70	52.51	**69.52**	**73.22**	**75.48**	91.75	77.97	63.68	19.02	18.34	18.43	83.57	68.61	59.10
Parabel	36.36	44.04	48.29	81.73	68.78	57.44	50.93	64.00	72.08	93.03	**79.16**	**64.52**	11.66	12.73	13.68	84.31	72.57	63.39
PLT	36.00	43.30	47.31	81.77	68.33	57.15	50.02	63.15	71.24	**93.37**	78.90	64.18	12.77	14.45	15.12	85.54	74.56	64.48
PS-PLT	44.73	48.52	50.84	79.19	67.81	57.15	66.81	72.05	74.88	88.04	77.16	63.84	**21.83**	**19.77**	**19.12**	74.12	65.87	59.08
	WikiLSHTC-325K, $A = 0.5, B = 0.4$						WikipediaLarge-500K, $A = 0.5, B = 0.4$						Amazon-670K, $A = 0.6, B = 2.6$					
ProXML	34.80	37.70	41.00	63.60	41.50	30.80	33.10	35.00	**39.40**	**68.80**	**48.90**	37.90	30.80	32.80	35.10	43.50	38.70	35.30
PW-DiSMEC	**37.12**	**40.36**	**43.57**	**65.27**	42.68	31.48	30.32	31.56	33.52	66.38	45.69	35.85	**31.24**	33.27	35.51	41.70	37.81	34.92
PfastreXML	30.66	31.55	33.12	56.05	36.79	27.09	29.20	27.60	27.70	59.50	40.20	30.70	29.30	30.80	32.43	39.46	35.81	33.05
Parabel	26.76	33.27	37.36	65.04	**43.23**	**32.05**	28.80	31.90	34.60	67.50	48.70	37.70	25.43	29.43	32.85	44.89	39.80	36.00
PLT	26.00	31.93	35.62	63.87	42.25	31.34	26.28	30.93	34.15	67.50	48.26	37.74	26.31	30.22	33.83	**45.01**	**40.21**	**36.72**
PS-PLT	32.84	36.17	39.20	64.57	43.17	32.01	**34.12**	**35.70**	38.14	67.53	48.68	**38.23**	31.14	**33.45**	**35.60**	43.71	39.72	36.60

Table 2: PS-PLT and PLT average CPU train and prediction time compared to other state-of-the-art algorithms.

Dataset	ProXML	PW-DiSMEC	PfastreXML	PLT	PS-PLT
	t_{train} [h]				
WikiLSHTC-325K	≈ 151760	≈ 1437	6.25	9.21	
WikipediaLarge-500K	≈ 1595920	≈ 16272	51.07	46.17	
Amazon-670K	≈ 75160	≈ 810	3.01	1.92	
	t_{test}/N_{test} [ms]				
WikiLSHTC-325K	≈ 90	≈ 82	4.10	4.96	12.40
WikipediaLarge-500K	≈ 496	≈ 457	15.24	26.40	60.01
Amazon-670K	≈ 111	≈ 103	9.96	12.06	20.40

We modified the recently introduced napkinXC [13] implementation of PLTs, [2] which obtains state-of-the-art results and uses the Uniform-Cost Search as its inference method. We train binary models in both BR and PLTs using the LIBLINEAR library [10] with L_2-regularized logistic regression. For PLTs, we use an ensemble of 3 trees built with the hierarchical 2-means clustering algorithm (with clusters of size 100), popularized by Parabel [18]. Because the tree-building procedure involves randomness, we repeat all PLTs experiments five times and report the mean performance. We report standard errors along with additional results for popular L_2-regularized squared hinge loss and for beam search variant of PS-PLT in the Appendix. The experiments were performed on an Intel Xeon E5-2697 v3 2.6GHz machine with 128GB of memory.

The main results of the experimental comparison are presented in Table 1. Propensity-scored BR and PLTs consistently obtain better propensity-scored precision@k. At the same time, they slightly drop the performance on the standard precision@k on four and improve it on two datasets. There is no single method that dominates others on all datasets, but PS-PLTs is the best sub-linear method, achieving best results on *psp*@{1, 3, 5} in this category on five out of six datasets, at the same time in many cases being competitive to ProXML and PW-DiSMEC that often require orders of magnitude

[2] Repository with the code and scripts to reproduce the experiments: https://github.com/mwydmuch/napkinXC

more time for training and prediction than PS-PLT. In Table 2, we show CPU train and test times of PS-PLTs compared to vanilla PLTs, PfasterXML, ProXML and PW-DiSMEC on our hardware (approximated for the last two using a subset of labels).

6 CONCLUSIONS

In this work, we demonstrated a simple approach for obtaining Bayes optimal predictions for propensity-scored precision@k, which can be applied to a wide group of probabilistic classifiers. Particularly we introduced an admissible and consistent inference algorithm for probabilistic labels trees, being the underlying model of such methods like Parabel, Bonsai, napkinXC, extremeText, AttentionXML and PECOS.

PS-PLTs show significant improvement with respect to propensity-scored precision@k, achieving state-of-the-art results in the group of algorithms with sub-linear training and prediction times. Furthermore, the introduced approach does not require any retraining of underlining classifiers if the propensities change. Since in real-world applications estimating true propensities may be hard, this property makes our approach suitable for dynamically changing environments, especially if we take into account the fact that many of PLTs-based algorithms can be trained incrementally [12, 14, 24, 25].

ACKNOWLEDGMENTS

Computational experiments have been performed in Poznan Supercomputing and Networking Center.

REFERENCES

[1] Shivani Agarwal. 2014. Surrogate regret bounds for bipartite ranking via strongly proper losses. *Journal of Machine Learning Research* 15, 1 (2014), 1653–1674.

[2] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. International World Wide Web Conferences Steering Committee / ACM, 13–24.

[3] Rohit Babbar and Bernhard Schölkopf. 2017. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. ACM, 721–729.

[4] Rohit Babbar and Bernhard Schölkopf. 2019. Data scarcity, robustness and extreme multi-label classification. *Machine Learning* 108 (09 2019). https://doi.org/10.1007/s10994-019-05791-5

[5] Alina Beygelzimer, John Langford, Yury Lifshits, Gregory B. Sorkin, and Alexander L. Strehl. 2009. Conditional Probability Tree Estimation Analysis and Algorithms. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. AUAI Press, 51–58.

[6] K. Bhatia, K. Dahiya, H. Jain, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code. http://manikvarma.org/downloads/XC/XMLRepository.html

[7] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. 2020. Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 3163–3171. https://dl.acm.org/doi/10.1145/3394486.3403368

[8] Ofer Dekel and Ohad Shamir. 2010. Multiclass-Multilabel Classification with More Classes than Examples. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010 (JMLR Proceedings)*, Vol. 9. JMLR.org, 137–144.

[9] Jia Deng, Sanjeev Satheesh, Alexander C. Berg, and Fei-Fei Li. 2011. Fast and Balanced: Efficient Label Tree Learning for Large Scale Object Recognition. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. 567–575.

[10] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.

[11] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme Multi-Label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 935–944. https://doi.org/10.1145/2939672.2939756

[12] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hüllermeier. 2016. Extreme F-measure Maximization using Sparse Probability Estimates. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings)*, Vol. 48. JMLR.org, 1435–1444.

[13] Kalina Jasinska-Kobus, Marek Wydmuch, Krzysztof Dembczyński, Mikhail Kuznetsov, and Róbert Busa-Fekete. 2020. Probabilistic Label Trees for Extreme Multi-Label Classification. *CoRR* abs/2009.11218 (2020).

[14] Kalina Jasinska-Kobus, Marek Wydmuch, Devanathan Thiruvenkatachari, and Krzysztof Dembczynski. 2021. Online probabilistic label trees. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Arindam Banerjee and Kenji Fukumizu (Eds.), Vol. 130. PMLR, 1801–1809. http://proceedings.mlr.press/v130/wydmuch21a.html

[15] Sujay Khandagale, Han Xiao, and Rohit Babbar. 2019. Bonsai - Diverse and Shallow Trees for Extreme Multi-label Classification. *CoRR* abs/1904.08249 (2019).

[16] Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. 2013. Beam search algorithms for multilabel learning. *Machine Learning* 92 (2013), 65–89.

[17] Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. 2019. Extreme Classification in Log Memory using Count-Min Sketch: A Case Study of Amazon Search with 50M Products. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 13265–13275. http://papers.nips.cc/paper/9482-extreme-classification-in-log-memory-using-count-min-sketch-a-case-study-of-amazon-search-with-50m-products.pdf

[18] 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 on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 993–1002.

[19] Yashoteja Prabhu and Manik Varma. 2014. FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. ACM, 263–272.

[20] Mohammadreza Qaraei, Erik Schultheis, Priyanshu Gupta, and Rohit Babbar. 2021. Convex Surrogates for Unbiased Loss Functions in Extreme Classification With Missing Labels. In *Proceedings of The Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3442381.3450139

[21] Stuart J. Russell and Peter Norvig. 2009. *Artificial Intelligence: a modern approach* (3 ed.). Pearson.

[22] Tong Wei and Yu-Feng Li. 2018. Does Tail Label Help for Large-Scale Multi-Label Learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (Stockholm, Sweden) *(IJCAI'18)*. AAAI Press, 2847–2853.

[23] Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label Partitioning For Sublinear Ranking. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013 (JMLR Workshop and Conference Proceedings)*, Vol. 28. JMLR.org, 181–189.

[24] Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. 2018. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 6355–6366.

[25] 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 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 5812–5822.

[26] Hsiang-Fu Yu, Kai Zhong, and Inderjit S Dhillon. 2020. PECOS: Prediction for Enormous and Correlated Output Spaces. *arXiv preprint arXiv:2010.05878* (2020).

[27] Tong Zhang. 2004. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Statist.* 32, 1 (02 2004), 56–85. https://doi.org/10.1214/aos/1079120130

[28] Jingwei Zhuo, Ziru Xu, Wei Dai, Han Zhu, Han Li, Jian Xu, and Kun Gai. 2020. Learning Optimal Tree Models under Beam Search. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria.

A DATASETS

Table 3: The number of unique features, labels, examples in train and test splits, and the average number of true labels per example in the benchmark data sets and corresponding A, B parameters for empirical propensity modeling.

Dataset	$\dim \mathcal{X}$	$\dim \mathcal{Y}$ (m)	N_{train}	N_{test}	avg. $\lvert\mathcal{L}_{\boldsymbol{x}}\rvert$	A	B
EurLex-4K	5000	3993	15539	3809	5.31	0.55	1.5
AmazonCat-13K	203882	13330	1186239	306782	5.04	0.55	1.5
Wiki10-31K	101938	30938	14146	6616	18.64	0.55	1.5
WikiLSHTC-325K	1617899	325056	1778351	587084	3.19	0.5	0.4
WikipediaLarge-500K	2381304	501070	1813391	783743	4.77	0.5	0.4
Amazon-670K	135909	670091	490449	153025	5.45	0.6	2.6

B PS-PLT WITH BEAM SEARCH INFERENCE

Beam search is a greedy search method that on each level of the tree keeps only b nodes with the highest probability estimates and discards the rest. Therefore it may not find the actual top k labels and may suffer regret for precision@k [28], but it guarantees logarithmic time and performs prediction level-by-level, which allows for easier implementation and memory management in large models. In Algorithm 2 we present beam search variant of PS-PLT. The presented algorithm assumes that tree T is balanced.

Algorithm 2 PS-PLT.PredictTopLabelsWithBeamSearch$(T, H, \boldsymbol{q}, \boldsymbol{x}, k, b)$

$\hat{\boldsymbol{y}} = \mathbf{0}$, $q_{\max} = \max_{j \in \mathcal{L}} q_j$, $\mathcal{B} = \emptyset$, ▷ Initialize prediction $\hat{\boldsymbol{y}}$ vector to all zeros, $q_{\max}$ and a list $\mathcal{B}$
$g(r_T, \boldsymbol{x}) = -\log \hat{\tilde{\eta}}(\boldsymbol{x}, r_T)$ ▷ Calculate cost $g(r_T, \boldsymbol{x})$ for the tree
$\hat{f}(r_T, \boldsymbol{x}) = g(r_T, \boldsymbol{x}) + \log q_{\max} - \log \max_{j \in \mathcal{L}_{r_T}} q_j$ ▷ Calculate estimated cost $\hat{f}(r_T, \boldsymbol{x})$ for the tree root
$\mathcal{B}$.add($(r_T, g(r_T, \boldsymbol{x}), \hat{f}(r_T, \boldsymbol{x}))$ ▷ Add the tree root with cost $g(r_T, \boldsymbol{x})$ and estimation $\hat{f}(r_T, \boldsymbol{x})$ to the list
for $d = 0; d <$ depth of $T; d = d + 1$ **do** ▷ For each level of the tree T
  $\mathcal{B}' =$ SelectTopNodes$(\mathcal{B}, b)$ ▷ Select b nodes from $\mathcal{B}'$ with highest values of $\hat{f}(v', \boldsymbol{x})$
  $\mathcal{B} = \emptyset$ ▷ Initialize list of nodes of the next level of the tree
  for $(v, g(v, \boldsymbol{x}), \_) \in \mathcal{B}'$ **do** ▷ Iterate over elements on the list $\mathcal{B}'$ (nodes and corresponding probabilities)
    for $v' \in \text{Ch}(v)$ **do** ▷ For all child nodes
      $g(v', \boldsymbol{x}) = g(v, \boldsymbol{x}) - \log \hat{\tilde{\eta}}(\boldsymbol{x}, v')$ ▷ Compute $g(v', \boldsymbol{x})$ using $\hat{\tilde{\eta}}(v', \boldsymbol{x}) \in H$
      $\hat{f}(v', \boldsymbol{x}) = g(v', \boldsymbol{x}) + \log q_{\max} - \log \max_{j \in \mathcal{L}_{v'}} q_j$ ▷ Calculate estimation $\hat{f}(v', \boldsymbol{x})$
      $\mathcal{B}$.add($(v', g(v', \boldsymbol{x}), \hat{f}(v', \boldsymbol{x}))$) ▷ Add the node, computed cost $g(v', \boldsymbol{x})$, and estimation $\hat{f}(v', \boldsymbol{x})$ to the list $\mathcal{B}$
for $(v, \_, \_) \in$ SelectTopNodes$(\mathcal{B}, k)$ **do** $\hat{y}_v = 1$ ▷ Select k leaves from $\mathcal{B}$ with highest values of $\hat{f}(v', \boldsymbol{x})$ and set the corresponding labels in $\hat{\boldsymbol{y}}$
return $\hat{\boldsymbol{y}}$ ▷ Return the prediction vector

C DETAILED RESULTS OF DIFFERENT VARIANTS OF PS-PLTS

In Table 4 we report the detailed results of PLT with nodes trained using logistic loss (*log*) and squared hinge loss (h^2) and PS-PLT with A*-search (A^*) presented in Algorithm 1 as well as with beam search version (*beam*) presented in Algorithm 2. For beam search variant we use $b = 10$ which is default value in many popular implementations, since it provides good trade off between predictive and computational performance when predicting top-5 labels. All variants use the ensemble of 3 trees and the same tree structures, built with the hierarchical 2-means clustering algorithm (with clusters of size 100). This means that the difference between variants is only in learning node classifiers and inference (tree search) methods.

The results show that all variants of PL-PLTs consistently obtain better propensity-scored precision@k. PS-PLTs trained with logistic loss achieves greater improvement in terms of $psp@\{1, 3, 5\}$ over vanilla PLT than variant trained with squared hinge loss. While PS-PLTs trained with squared hinge loss suffer a small drop in the performance on the standard precision@k. For both losses, beam search variant allows for further decrease of inference time at the cost of an only minor decrease in terms of predictive performance.

Table 4: Mean performance with standard errors, rounded to two decimal places, of different variants of PS-PLTs on propensity-scored and standard precision@{1, 3, 5} [%], train time [h] and inference time per example [ms]. The best result for each measure is in bold.

	psp@1	psp@3	psp@5	p@1	p@3	p@5	T_{train}	T_{test}/N_{test}
	EurLex-4K							
PLT$_{log}$	36.00 ± 0.07	43.30 ± 0.09	47.31 ± 0.09	**81.77 ± 0.09**	68.33 ± 0.11	57.15 ± 0.08	0.04 ± 0.00	2.83 ± 0.10
PS-PLT$_{log+A^*}$	**44.73 ± 0.06**	**48.52 ± 0.11**	**50.84 ± 0.11**	79.19 ± 0.09	67.81 ± 0.07	57.15 ± 0.09		5.66 ± 0.14
PS-PLT$_{log+beam}$	44.72 ± 0.07	48.48 ± 0.11	50.77 ± 0.12	79.19 ± 0.09	67.78 ± 0.06	57.12 ± 0.11		1.75 ± 0.12
PLT$_{h^2}$	36.21 ± 0.05	44.01 ± 0.12	48.41 ± 0.16	81.66 ± 0.14	**68.75 ± 0.15**	**57.54 ± 0.14**	**0.02 ± 0.00**	1.83 ± 0.07
PS-PLT$_{h^2+A^*}$	44.21 ± 0.05	48.51 ± 0.12	50.60 ± 0.15	80.72 ± 0.14	67.99 ± 0.10	56.20 ± 0.14		2.60 ± 0.17
PS-PLT$_{h^2+beam}$	44.21 ± 0.08	48.49 ± 0.13	50.57 ± 0.13	80.69 ± 0.14	67.97 ± 0.12	56.22 ± 0.11		**1.65 ± 0.02**
	AmazonCat-13K							
PLT$_{log}$	50.02 ± 0.01	63.15 ± 0.03	71.24 ± 0.06	**93.37 ± 0.02**	78.90 ± 0.04	64.18 ± 0.05	3.14 ± 0.06	1.74 ± 0.07
PS-PLT$_{log+A^*}$	**66.81 ± 0.03**	**72.05 ± 0.04**	**74.88 ± 0.05**	88.04 ± 0.05	77.16 ± 0.04	63.84 ± 0.03		3.71 ± 0.37
PS-PLT$_{log+beam}$	66.78 ± 0.03	72.01 ± 0.04	74.85 ± 0.04	88.04 ± 0.05	77.16 ± 0.04	63.84 ± 0.03		1.19 ± 0.09
PLT$_{h^2}$	50.91 ± 0.01	63.89 ± 0.03	71.94 ± 0.06	93.00 ± 0.04	**79.06 ± 0.04**	**64.43 ± 0.04**	**1.01 ± 0.05**	1.20 ± 0.04
PS-PLT$_{h^2+A^*}$	65.97 ± 0.03	71.96 ± 0.06	74.76 ± 0.11	88.76 ± 0.05	77.75 ± 0.06	63.75 ± 0.08		2.31 ± 0.04
PS-PLT$_{h^2+beam}$	65.96 ± 0.03	71.94 ± 0.07	74.84 ± 0.12	88.77 ± 0.06	77.75 ± 0.06	63.84 ± 0.09		**0.92 ± 0.03**
	Wiki10-31K							
PLT$_{log}$	12.77 ± 0.02	14.45 ± 0.01	15.12 ± 0.01	**85.54 ± 0.07**	**74.56 ± 0.05**	**64.48 ± 0.03**	0.46 ± 0.01	25.08 ± 0.39
PS-PLT$_{log+A^*}$	**21.83 ± 0.07**	**19.77 ± 0.03**	**19.12 ± 0.04**	74.12 ± 0.09	65.87 ± 0.13	59.08 ± 0.15		74.37 ± 1.08
PS-PLT$_{log+beam}$	21.14 ± 0.06	19.02 ± 0.05	18.43 ± 0.07	74.33 ± 0.12	66.20 ± 0.23	59.62 ± 0.24		5.63 ± 0.05
PLT$_{h^2}$	11.68 ± 0.01	12.84 ± 0.02	13.79 ± 0.02	84.31 ± 0.13	72.90 ± 0.06	63.75 ± 0.04	**0.28 ± 0.01**	10.91 ± 0.13
PS-PLT$_{h^2+A^*}$	18.51 ± 0.02	17.61 ± 0.04	18.04 ± 0.05	83.06 ± 0.07	71.19 ± 0.14	62.66 ± 0.11		33.06 ± 0.70
PS-PLT$_{h^2+beam}$	18.33 ± 0.02	17.36 ± 0.04	17.65 ± 0.05	83.03 ± 0.07	71.17 ± 0.14	62.58 ± 0.13		**4.81 ± 0.21**
	WikiLSHTC-325K							
PLT$_{log}$	26.00 ± 0.08	31.93 ± 0.11	35.62 ± 0.13	63.87 ± 0.19	42.25 ± 0.13	31.34 ± 0.10	9.21 ± 0.10	4.96 ± 0.16
PS-PLT$_{log+A^*}$	32.84 ± 0.18	36.27 ± 0.24	**39.38 ± 0.27**	64.47 ± 0.43	**43.19 ± 0.29**	**32.08 ± 0.21**		12.40 ± 0.74
PS-PLT$_{log+beam}$	32.76 ± 0.18	36.09 ± 0.25	39.08 ± 0.31	64.38 ± 0.44	43.05 ± 0.30	31.91 ± 0.23		1.21 ± 0.04
PLT$_{h^2}$	26.71 ± 0.08	33.14 ± 0.16	37.06 ± 0.22	64.69 ± 0.18	42.95 ± 0.16	31.82 ± 0.15	**6.35 ± 0.10**	2.31 ± 0.06
PS-PLT$_{h^2+A^*}$	**33.16 ± 0.12**	**36.39 ± 0.29**	38.14 ± 0.42	**65.87 ± 0.23**	42.68 ± 0.27	30.46 ± 0.26		5.00 ± 0.64
PS-PLT$_{h^2+beam}$	33.11 ± 0.12	36.23 ± 0.28	37.92 ± 0.39	65.82 ± 0.23	42.58 ± 0.27	30.41 ± 0.25		**1.13 ± 0.06**
	WikipediaLarge-500K							
PLT$_{log}$	26.28 ± 0.09	30.93 ± 0.12	34.15 ± 0.14	67.50 ± 0.27	48.26 ± 0.20	37.74 ± 0.15	46.17 ± 0.32	26.40 ± 0.64
PS-PLT$_{log+A^*}$	**34.12 ± 0.10**	**35.70 ± 0.12**	**38.14 ± 0.13**	67.53 ± 0.21	48.68 ± 0.15	38.23 ± 0.12		60.01 ± 5.58
PS-PLT$_{log+beam}$	34.11 ± 0.11	35.65 ± 0.13	38.06 ± 0.15	67.54 ± 0.23	48.63 ± 0.17	38.17 ± 0.14		**4.81 ± 0.13**
PLT$_{h^2}$	26.71 ± 0.08	31.62 ± 0.14	34.91 ± 0.19	68.28 ± 0.23	**49.13 ± 0.20**	**38.33 ± 0.18**	**27.05 ± 0.19**	10.21 ± 0.07
PS-PLT$_{h^2+A^*}$	33.77 ± 0.11	35.64 ± 0.21	37.37 ± 0.31	68.54 ± 0.23	48.45 ± 0.26	37.03 ± 0.27		21.20 ± 4.31
PS-PLT$_{h^2+beam}$	33.75 ± 0.15	35.60 ± 0.29	37.33 ± 0.41	**68.57 ± 0.32**	48.44 ± 0.36	37.07 ± 0.36		4.91 ± 0.37
	Amazon-670K							
PLT$_{log}$	26.31 ± 0.06	30.22 ± 0.08	33.83 ± 0.10	**45.01 ± 0.12**	**40.21 ± 0.11**	**36.72 ± 0.10**	1.92 ± 0.01	12.06 ± 0.05
PS-PLT$_{log+A^*}$	**31.14 ± 0.07**	**33.45 ± 0.09**	**35.60 ± 0.11**	43.71 ± 0.10	39.72 ± 0.09	36.60 ± 0.10		20.40 ± 0.45
PS-PLT$_{log+beam}$	30.95 ± 0.07	33.13 ± 0.11	35.14 ± 0.14	43.48 ± 0.11	39.40 ± 0.11	36.21 ± 0.13		1.57 ± 0.15
PLT$_{h^2}$	26.22 ± 0.08	29.89 ± 0.12	33.12 ± 0.16	44.78 ± 0.17	39.75 ± 0.16	35.97 ± 0.16	**1.44 ± 0.01**	4.56 ± 0.14
PS-PLT$_{h^2+A^*}$	29.92 ± 0.09	32.23 ± 0.12	34.21 ± 0.17	43.57 ± 0.15	38.95 ± 0.13	35.33 ± 0.16		6.59 ± 0.04
PS-PLT$_{h^2+beam}$	29.82 ± 0.09	32.02 ± 0.12	33.91 ± 0.17	43.45 ± 0.15	38.75 ± 0.14	35.08 ± 0.16		**1.17 ± 0.04**