A Training in PLT

The pseudocode with a brief description of the training algorithm for IPLTs is given in the main text. Here we discuss it in more detail. The algorithm first initializes all node classifiers. Training relies on a proper assignment of training examples to nodes. To train probabilistic classifiers $\hat{\eta}(v)$, $v \in V_T$, in all nodes of a tree T, we need to properly filter training examples as given in (2). In each iteration the algorithm identifies for a given training example a set of *positive* and *negative nodes*, i.e., the nodes for which a training example is treated respectively as positive (i.e, $(\boldsymbol{x}, z_v = 1)$), or negative (i.e., $(\boldsymbol{x}, z_v = 0)$). The AssignToNodes method is given in Algorithm 9. It initializes the positive nodes to the empty set and the negative nodes to the root node (to deal with $\boldsymbol{y}$ of all zeros). Next, it traverses the tree from the leaves corresponding to the labels of the training example to the root adding the visited nodes to the set of positive nodes. It also removes each visited node from the set of negative nodes, if it has been added to this set before. All children of the visited node, which are not in the set of positive nodes, are then added to the set of negative nodes. If the parent node of the visited node has already been added to positive nodes, the traversal on this path stops.

Using the positive and negative nodes the procedure updates the corresponding classifiers with an online learner A_{online} of choice. Note that all node classifiers are trained in fact independently as updates in any node do not influence training of the other nodes. The output of the algorithm is a set of probabilistic classifiers H.

Algorithm 9 IPLT/OPLT.AssignToNodes$(T, \boldsymbol{x}, \mathcal{L}_{\boldsymbol{x}})$

```
1:  $P = \emptyset$, $N = \{r_T\}$                    ▷ Initialize sets of positive and negative nodes
2:  for $j \in \mathcal{L}_{\boldsymbol{x}}$ do                      ▷ For all labels of the training example
3:      $v = \ell_j$                                 ▷ Set $v$ to a leaf corresponding to label $j$
4:      while $v$ not null and $v \notin P$ do    ▷ On a path to the root or the first positive node (excluded)
5:          $P = P \cup \{v\}$                         ▷ Assign a node to positive nodes
6:          $N = N \setminus \{v\}$                     ▷ Remove the node from negative nodes if added there before
7:          for $v' \in \mathrm{Ch}(v)$ do                 ▷ For all its children
8:              if $v' \notin P$ then                    ▷ If a child is not a positive node
9:                  $N = N \cup \{v'\}$                 ▷ Assign it to negative nodes
10:         $v = \mathrm{pa}(v)$                         ▷ Move up along the path
11: return $(P, N)$                       ▷ Return a set of positive and negative nodes for the training example
```

B Prediction in PLT

Algorithm 10 outlines the prediction procedure for PLTs that returns top k labels. It is based on the uniform-cost search. Alternatively, one can use beam search.

Algorithm 10 IPLT/OPLT.PredictTopLabels$(T, H, k, \boldsymbol{x})$

```
1:  $\hat{\boldsymbol{y}} = \boldsymbol{0}$, $\mathcal{Q} = \emptyset$,   ▷ Initialize prediction vector to all zeros and a priority queue, ordered descending by $\hat{\eta}_v(\boldsymbol{x})$
2:  $k' = 0$                                   ▷ Initialize counter of predicted labels
3:  $\mathcal{Q}$.add$((r_T, \hat{\eta}(\boldsymbol{x}, r_T)))$         ▷ Add the tree root with the corresponding estimate of probability
4:  while $k' < k$ do                          ▷ While the number of predicted labels is less than $k$
5:      $(v, \hat{\eta}_v(\boldsymbol{x})) = \mathcal{Q}$.pop()                ▷ Pop the top element from the queue
6:      if $v$ is a leaf then                       ▷ If the node is a leaf
7:          $\hat{y}_v = 1$                          ▷ Set the corresponding label in the prediction vector
8:          $k' = k' + 1$                            ▷ Increment the counter
9:      else                                       ▷ If the node is an internal node
10:         for $v' \in \mathrm{Ch}(v)$ do                  ▷ For all child nodes
11:             $\hat{\eta}_{v'}(\boldsymbol{x}) = \hat{\eta}_v(\boldsymbol{x}) \times \hat{\eta}(\boldsymbol{x}, v')$    ▷ Compute $\hat{\eta}_{v'}(\boldsymbol{x})$ using $\hat{\eta}(v') \in H$
12:             $\mathcal{Q}$.add$((v', \hat{\eta}_{v'}(\boldsymbol{x})))$     ▷ Add the node and the computed probability estimate
13: return $\hat{\boldsymbol{y}}$                              ▷ Return the prediction vector
```

C The proof of the result from Section 4.3

Theorem 2 concerns two properties, properness and efficency, of an OPLT algorithm. We first prove that the OPLT algorithm satisfies each of the property in two separate lemmas. The final proof of the theorem is then straight-forward.

Lemma 1. OPLT is a proper OPLT algorithm.

Proof. We need to show that for any $\mathcal{S}$ and t the two of the following hold. Firstly, that the set L_{T_t} of leaves of tree T_t built by OPLT correspond to $\mathcal{L}_t$, the set of all labels observed in S_t. Secondly, that the set H_t of classifiers trained by OPLT is exactly the same as $H = \text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_t)$, i.e., the set of node classifiers trained incrementally by Algorithm 1 on $\mathcal{D} = \mathcal{S}_t$ and tree T_t given as input parameter. We will prove it by induction with the base case for $\mathcal{S}_0$ and the induction step for $\mathcal{S}_t$, $t \geq 1$, with the assumption that the statement holds for $\mathcal{S}_{t-1}$.

For the base case of $\mathcal{S}_0$, tree T_0 is initialized with the root node r_T with no label assigned and set H_0 of node classifiers with a single classifier assigned to the root. As there are no observations, this classifier receives no updates. Now, notice that IPLT.Train, run on T_0 and $\mathcal{S}_0$, returns exactly the same set of classifiers H that contains solely the initialized root node classifier without any updates (assuming that initialization procedure is always the same). There are no labels in any sequence of 0 observations and also T_0 has no label assigned.

The induction step is more involved as we need to take into account the internal loop which extends the tree with new labels. Let us consider two cases. In the first one, observation $(\boldsymbol{x}_t, \mathcal{L}_{\boldsymbol{x}_t})$ does not contain any new label. This means that that the tree T_{t-1} will not change, i.e., $T_{t-1} = T_t$. Moreover, node classifiers from H_{t-1} will get the same updates for $(\boldsymbol{x}_t, \mathcal{L}_{\boldsymbol{x}_t})$ as classifiers in IPLT.Train, therefore $H_t = \text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_t)$. It also holds that $l_j \in L_{T_t}$ iff $j \in \mathcal{L}_t$, since $\mathcal{L}_{t-1} = \mathcal{L}_t$. In the second case, observation $(\boldsymbol{x}_t, \mathcal{L}_{\boldsymbol{x}_t})$ has $m' = |\mathcal{L}_{\boldsymbol{x}_t} \setminus \mathcal{L}_{t-1}|$ new labels. Let us make the following assumption for the UpdateTree procedure, which we later prove that it indeed holds. Namely, we assume that the set $H_{t'}$ of classifiers after calling the UpdateTree procedure is the same as the one being returned by $\text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_{t-1})$, where T_t is the extended tree. Moreover, leaves of T_t correspond to all observed labels seen so far. If this is the case, the rest of the induction step is the same as in the first case. All updates to classifiers in $H_{t'}$ for $(\boldsymbol{x}_t, \mathcal{L}_{\boldsymbol{x}_t})$ are the same as in IPLT.Train. Therefore $H_t = \text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_t)$.

Now, we need to show that the assumption for the UpdateTree procedure holds. To this end we also use induction, this time on the number m' of new labels. For the base case, we take $m' = 1$. The induction step is proved for $m' > 1$ with the assumption that the statement holds for $m' - 1$.

For $m' = 1$ we need consider two scenarios. In the first scenario, the new label is the first label in the sequence. This label will be then assigned to the root node r_T. So, the structure of the tree does not change, i.e., $T_{t-1} = T_t$. Furthermore, the set of classifiers also does not changed, since the root classifier has already been initialized. It might be negatively updated by previous observations. Therefore, we have $H_{t'} = \text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_{t-1})$. Furthermore, all observed labels are appropriately assigned to the leaves of T_t. In the second scenario, set $\mathcal{L}_{t-1}$ is not empty. We need to consider in this scenario the three variants of tree extension illustrated in Figure 1.

In the first variant, tree T_{t-1} is extended by one leaf node only without any additional ones. AddNode creates a new leaf node v'' with the new label assigned to the tree. After this operation the tree contains all labels from $\mathcal{S}_t$. The new leaf v'' is added as a child of the selected node v. This new node is initialized as $\hat{\eta}(v'') = \text{InverseClassifier}(\hat{\theta}(v))$. Recall that InverseClassifier creates a wrapper that inverts the behavior of the base classifier. It predicts $1 - \hat{\eta}$, where $\hat{\eta}$ is the prediction of the base classifier, and flips the updates, i.e., positive updates become negative and negative updates become positive. From the definition of the auxiliary classifier, we know that $\hat{\theta}(v)$ has been trained on all positives updates of $\hat{\eta}(v)$. So, $\hat{\eta}(v'')$ is initialized with a state as if it was updated negatively each time $\hat{\eta}(v)$ was updated positively in sequence S_{t-1}. Notice that in S_{t-1} there is no observation labeled with the new label. Therefore $\hat{\eta}(v'')$ is the same as if it was created and updated using IPLT.Train. There are no other operations on T_{t-1}, so we have that $H_{t'} = \text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_{t-1})$.

In the second variant, tree T_{t-1} is extended by internal node v' and leaf node v''. The internal node v' is added in InsertNode. It becomes a parent of all child nodes of the selected node v and the only child of this node. Thus, all leaves of the subtree of v does not change. Since v' is the root of this subtree, its classifier $\hat{\eta}(v')$ should be initialized as a copy of the auxiliary classifier $\hat{\theta}(v)$, which has accumulated all updates from and only from observations with labels assigned to the leaves of this subtree. Addition of the leaf node v'' can be analyzed as in the first variant. Since nothing else has changed in the tree and in the node classifiers, we have that $H_{t'} = \text{IPLT.Train}(T_t, A_{\text{online}}, \mathcal{S}_{t-1})$. Moreover, the tree contains the new label, so the statement holds.

The third variant is similar to the second one. Tree T_{t-1} is extended by two leaf nodes v' and v'' being children of the selected node v. Insertion of leaf v' is similar to insertion of node v' in the second variant, with the difference that v does not have any children and its label has to be reassigned to v'. The new classifier in v' is initialized as a copy of the auxiliary classifier $\hat{\theta}(v)$, which contains all updates from and only from observations with the label assigned previously to v. Insertion of v'' is exactly the same as in the second variant. From the above, we conclude that $H_{t'} = \text{IPLT}.\textsc{Train}(T_t, A_{\text{online}}, \mathcal{S}_{t-1})$ and that T_t contains all labels from T_{t-1} and the new label. In this way we prove the base case.

The induction step is similar to the second scenario of the base case. The only difference is that we do not extent tree T_{t-1}, but an intermediate tree with $m' - 1$ new labels already added. Because of the induction hypothesis, the rest of the analysis of the three variants of tree extension is exactly the same. This ends the proof that the assumption for the inner loop holds. At the same time it finalizes the entire proof. □

Lemma 2. OPLT is an efficient OPLT algorithm.

Proof. The OPLT maintains one additional classifier per each node in comparison to IPLT. Hence, for a single observation there is at most one update more for each positive node. Furthermore, the time and space cost of the complete tree building policy is constant per a single label, if implemented with an array list. In this case, insertion of any new node can be made in amortized constant time, and the space required by the array list is linear in the number of nodes. Concluding the above, the time and space complexity of OPLT is in constant factor of C_t and C_s, the time and space complexity of IPLT respectively. This proves that OPLT is an efficient OPLT algorithm. □

Theorem 2. OPLT is a proper and efficient online PLT algorithm.

Proof. The theorem directly follows from Lemma 1 and Lemma 2. □

D Detailed results of OPLT and CMT on extreme multi-label classification tasks

In Table 4, complementary to Table 2, we report performance on propensity scored precision at $\{1, 3, 5\}$ (Jain et al., 2016) defined as:

$$\text{PSP@}k = \frac{1}{k} \sum_{j \in \hat{\mathcal{L}}_{\boldsymbol{x}}} q_j [\![\tilde{y}_j = 1]\!] \,,$$

where $q_j = 1 + C(N_j + B)^{-A}$ is inverse propensity of label j, where N_j is the number of data points annotated with label j in the observed ground truth dataset of size N, A, B are dataset specific parameters and $C = (\log N - 1)(B + 1)^A$. For Wiki10 and AmazonCat datasets we use $A = 0.55, B = 1.5$, for WikiLSHTC $A = 0.5, B = 0.4$, and for Amazon $A = 0.6, B = 2.6$ as recommended in (Jain et al., 2016). Since values of q_j are higher for infrequent labels, propensity scored precision at k promotes the accurate prediction on harder to predict tail labels. As in the case of the results in terms of standard precision at k, OPLT outperforms CMT, being slightly worse than its offline counterparts IPLT and Parabel, especially on the WikiLSHTC dataset.

Table 4: Mean propensity weighted precision at $\{1, 3, 5\}$ (%) of Parabel, IPLT, CMT, OPLT for XMLC tasks. Notation: PP@k – propensity weighted precision at k-position, R – Random policy, B – Best-greedy policy.

	AmazonCat			Wiki10			WikiLSHTC			Amazon		
Algorithm	PP@1	PP@3	PP@5	PP@1	PP@3	PP@5	PP@1	PP@3	PP@5	PP@1	PP@3	PP@5
Parabel	50.89	63.60	71.27	11.73	12.70	13.68	25.91	31.50	34.77	25.39	28.57	31.21
IPLT	49.97	63.13	70.77	12.26	13.75	14.80	24.16	29.09	32.29	25.25	28.85	32.12
CMT	47.48	54.25	56.03	9.68	9.66	9.67	-	-	-	-	-	-
OPLT$_R$	48.98	60.67	67.51	11.64	13.07	14.12	16.12	19.70	22.68	20.28	24.04	27.35
OPLT$_B$	49.30	61.55	68.64	11.59	13.19	14.26	20.69	24.87	28.14	23.41	27.26	30.49
OPLT-W	49.86	62.91	70.44	11.69	13.41	14.43	22.56	27.10	30.23	23.65	27.62	30.95

In Table 5, we show detailed results of the empirical study we performed to evaluate OPLTs comprehensively. In addition to precision at $\{1, 3, 5\}$ and train times, we also report test times. We present the results for every algorithm that is using an incremental learning algorithm, updating the tree nodes (IPLT, CMT, and OPLT) after 1 and 3 passes over the training dataset. We also present the results of OPLT-W with a warm-start sample of size 5, 10, and 15% of the training dataset (warm-up dataset). In all cases, the initial tree was created using hierarchical 2-means clustering on the warm-up sample and later extended using Best-greedy policy. The results show that IPLT and OPLT achieve good results after just one pass over datasets. Prediction times of OPLT are slightly worse than IPLT, probably due to the worst tree structure, the prediction times of OPLT with warm-start are better than OPLT build from scratch. The reported prediction times of Parabel are given for the batch prediction. The prediction times seem to be faster, but Parabel needs to decompress the model during prediction, which makes it less suitable for online prediction. It is only efficient when the batches are sufficiently large.

Table 5: Mean precision at $\{1,3,5\}$ (%), train and test CPU time (averaged over 5 runs) of Parabel, IPLT, CMT, OPLT for XMLC tasks. Notation: P@k – precision at k-position, T – CPU time, R – Random policy, B – Best-greedy policy, w – percentage of the training dataset sampled for warm-start, p – number of passes over train dataset, $*$ – offline prediction, depends on the test dataset size. Highlighted values are presented in the main paper in Table 2.

	AmazonCat					Wiki10				
Algorithm	P@1	P@3	P@5	t_{train}	t_{test}/N	P@1	P@3	P@5	t_{train}	t_{test}/N
Parabel	92.58	78.53	63.90	11.51m	*0.3ms	84.17	72.12	63.30	4.00m	*0.9ms
IPLT$_{p=1}$	93.11	78.72	63.98	34.2m	0.6ms	84.87	74.42	65.31	18.3m	10.1ms
IPLT$_{p=3}$	93.19	78.64	63.92	115.7m	0.6ms	84.37	73.90	64.70	46.8m	10.4ms
CMT$_{p=1}$	87.51	69.49	53.99	45.8m	1.1ms	80.59	64.17	55.05	35.1m	16.5ms
CMT$_{p=3}$	89.43	70.49	54.23	168.3m	1.5ms	79.86	64.72	55.25	120.9m	20.4ms
OPLT$_{R,p=1}$	92.66	77.44	62.52	99.5m	1.7ms	84.34	73.73	64.41	30.3m	14.1ms
OPLT$_{R,p=3}$	92.65	77.43	62.59	278.3m	1.4ms	83.09	72.25	63.33	86.5m	18.2ms
OPLT$_{B,p=1}$	92.74	77.74	62.91	84.1m	1.5ms	84.47	73.73	64.39	27.7m	16.4ms
OPLT$_{B,p=3}$	92.77	77.70	62.93	251.9m	1.4ms	83.39	72.50	63.48	86.7m	18.2ms
OPLT-W$_{w=5\%,p=1}$	93.14	78.67	63.85	40.6m	0.8ms	85.10	74.44	64.77	28.7m	16.6ms
OPLT-W$_{w=5\%,p=3}$	93.11	78.50	63.75	134.3m	0.8ms	83.93	73.34	63.75	75.9m	17.0ms
OPLT-W$_{w=10\%,p=1}$	93.14	78.68	63.92	43.7m	0.9ms	85.22	74.68	64.93	28.2m	17.8ms
OPLT-W$_{w=10\%,p=3}$	93.10	78.51	63.79	133.9m	0.8ms	84.75	74.02	64.33	83.9m	18.3ms
OPLT-W$_{w=15\%,p=1}$	93.17	78.75	63.95	50.1m	1.0ms	85.42	74.50	64.94	28.2m	17.5ms
OPLT-W$_{w=15\%,p=3}$	93.15	78.54	63.82	121.6m	0.8ms	84.69	73.90	64.29	79.1m	19.5ms
	WikiLSHTC					Amazon				
Algorithm	P@1	P@3	P@5	t_{train}	t_{test}/N	P@1	P@3	P@5	t_{train}	t_{test}/N
Parabel	62.78	41.22	30.27	15.4m	*0.3ms	43.13	37.94	34.00	7.6m	*0.3ms
IPLT$_{p=1}$	58.14	37.94	28.14	69.0m	2.1ms	40.78	35.88	32.28	33.2m	5.1ms
IPLT$_{p=3}$	60.80	39.58	29.24	175.1m	2.6ms	43.55	38.69	35.20	79.4m	6.2ms
OPLT$_{R,p=1}$	46.36	29.85	22.53	103.0m	6.4ms	36.27	32.13	29.01	46.1m	17.0ms
OPLT$_{R,p=3}$	47.76	30.97	23.37	330.1m	7.9ms	38.42	34.33	31.32	134.2m	18.2ms
OPLT$_{B,p=1}$	53.32	34.22	25.52	88.6m	4.8ms	38.42	34.03	30.73	36.7m	10.0ms
OPLT$_{B,p=3}$	54.69	35.32	26.31	300.0m	5.8ms	41.09	36.65	33.42	111.9m	13.0ms
OPLT-W$_{w=5\%,p=1}$	57.46	36.95	27.35	82.5m	3.6ms	39.45	34.85	31.44	35.3m	9.1ms
OPLT-W$_{w=5\%,p=3}$	58.51	37.92	28.04	249.8m	4.6ms	41.96	37.39	34.07	100.0m	11.1ms
OPLT-W$_{w=10\%,p=1}$	58.20	37.50	27.73	71.9m	3.0ms	39.71	35.03	31.57	36.6m	9.1ms
OPLT-W$_{w=10\%,p=3}$	59.23	38.39	28.38	205.7m	3.4ms	42.21	37.60	34.25	98.3m	11.3ms
OPLT-W$_{w=15\%,p=1}$	58.68	37.85	27.97	72.1m	2.8ms	40.11	35.35	31.84	34.0m	8.3ms
OPLT-W$_{w=15\%,p=3}$	59.66	38.67	28.57	196.0m	3.1ms	42.41	37.70	34.29	85.1m	9.1ms

E Detailed results of OPLT and CMT on few-shot multi-class classification tasks

In Table 6 we present additional results for experiments on few-shot multi-class classification. We report results on the test set after 1 and 3 passes. In addition to accuracy and train times, we also report test times after 1 and 3 passes over the training dataset.

Table 6: Mean accuracy of prediction (%) and train and test CPU time of CMT, OPLT for few-shot multi-class classification tasks. Notation: Acc – accuracy, T – CPU time, N – number of samples in test set, R – RANDOM policy, B – BEST-GREEDY policy, p – number of passes over train dataset. Highlighted values are presented in the main paper in Table 3.

	ALOI			Wikipara 1-shot			Wikipara 3-shot			Wikipara 5-shot		
Algorithm	Acc.	t_{train}	t_{test}/N	Acc.	t_{train}	t_{test}/N	Acc.	t_{train}	t_{test}/N	Acc.	t_{train}	t_{test}/N
CMT$_{p=1}$	17.63	37.8s	0.78ms	2.22	7.1s	0.51ms	3.22	20.9s	0.76ms	4.15	81.1s	3.22ms
OPLT$_{R,p=1}$	62.58	6.0s	0.12ms	1.51	2.6s	0.82ms	13.31	6.7s	1.83ms	26.06	11.6s	2.61ms
OPLT$_{B,p=1}$	63.91	6.1s	0.12ms	0.80	2.2s	0.55ms	11.40	6.4s	1.67ms	23.12	10.3s	2.48ms
CMT$_{p=3}$	71.98	207s	0.57ms	2.48	21.3s	0.37ms	3.28	63.1s	0.49ms	4.21	240.9s	1.22ms
OPLT$_{R,p=3}$	66.50	29.3s	0.11ms	8.99	4.6s	1.14ms	27.34	16.4s	2.64ms	40.67	27.6s	2.83ms
OPLT$_{B,p=3}$	67.26	18.1s	0.10ms	3.31	4.4s	0.87ms	24.66	15.6s	2.37ms	39.13	27.2s	2.54ms

F Performance of OPLT with BEST-GREEDY policy and different α values

In Table 7, OPLT with BEST-GREEDY policy and different values of tree balancing parameter α is compared to IPLT and OPLT with RANDOM policy. All other parameters of the model were set as described in Section 5.1. It shows that for $\alpha \geq 0.75$, OPLT achieves the tree depth close or the same as perfectly balanced tree build for IPLT, at the same time having the best predictive performance and the shortest training and prediction time among OPLT variants.

G Performance analysis of OPLT with warm-start

In Table 8, we compare OPLT with warm-start (OPLT-W) with two variants of IPLT. In the first variant, IPLT is trained only on the warm-up training dataset, and in the second variant, IPLT uses a tree created on warm-up, but then updated with all examples from the training set but without updating the tree structure. In both variants, IPLT cannot predict labels that are not present in the initial warm-up dataset. All other parameters of the model were set as described in Section 5.1. This experiment shows the significant gain in predictive performance on WikiLSHTC and Amazon datasets by extending a tree with newly observed labels over the IPLT variants that do not take new labels into account.

Table 7: Mean precision at $\{1, 3, 5\}$ (%), train time, test time and tree depth (averaged over 5 runs) of OPLT with Best-greedy policy with different α values compared with IPLT and OPLT with Random policy. Notation: P@k – precision at k-position, t – CPU time, d(T) – tree depth, R – Random policy, B – Best-greedy policy, α – tree balancing parameter, percentage of the training dataset sampled for warm-start, p – number of passes over train dataset.

	AmazonCat						Wiki10					
Algorithm	P@1	P@3	P@5	t_{train}	t_{test}/N	d(T)	P@1	P@3	P@5	t_{train}	t_{test}/N	d(T)
IPLT$_{p=3}$	93.10	78.52	63.81	115.7m	0.6ms	10.0	83.49	73.06	64.12	46.8m	10.4ms	11.0
OPLT$_{R,p=3}$	92.65	77.43	62.59	278.3m	1.4ms	10.0	83.09	72.25	63.33	86.5m	18.2ms	11.0
OPLT$_{B,\alpha=0.25,p=3}$	92.77	77.77	62.99	303.1m	1.5ms	30.0	82.93	72.41	63.38	88.2m	20.0ms	24.6
OPLT$_{B,\alpha=0.375,p=3}$	92.80	77.72	62.93	250.2m	1.5ms	18.0	83.10	72.43	63.37	84.6m	18.9ms	17.2
OPLT$_{B,\alpha=0.5,p=3}$	92.79	77.74	62.95	257.6m	1.4ms	14.0	83.09	72.48	63.45	86.0m	18.7ms	14.0
OPLT$_{B,\alpha=0.625,p=3}$	92.77	77.74	62.96	245.2m	1.4ms	12.0	83.25	72.50	63.48	87.3m	19.6ms	12.6
OPLT$_{B,\alpha=0.75,p=3}$	92.77	77.70	62.93	248.9m	1.4ms	11.0	83.39	72.50	63.48	86.7m	18.2ms	12.0
OPLT$_{B,\alpha=0.875,p=3}$	92.73	77.64	62.85	207.7m	1.2ms	10.0	83.41	72.50	63.44	82.2m	17.6ms	11.0

	WikiLSHTC						Amazon					
Algorithm	P@1	P@3	P@5	t_{train}	t_{test}/N	d(T)	P@1	P@3	P@5	t_{train}	t_{test}/N	d(T)
IPLT$_{p=3}$	60.80	39.58	29.24	175.1m	2.6ms	14.0	43.55	38.69	35.20	79.4m	6.2ms	15.0
OPLT$_{R,p=3}$	47.76	30.97	23.37	330.1m	7.9ms	14.8	38.42	34.33	31.32	134.2m	18.2ms	16.0
OPLT$_{B,\alpha=0.25,p=3}$	53.14	34.36	25.69	337.8m	6.3ms	68.4	38.83	34.72	31.69	139.2m	16.4ms	38.4
OPLT$_{B,\alpha=0.375,p=3}$	53.47	34.57	25.81	309.1m	6.9ms	30.8	39.17	35.04	31.99	126.6m	16.2ms	21.2
OPLT$_{B,\alpha=0.5,p=3}$	53.77	34.74	25.93	294.3m	6.0ms	21.8	39.47	35.26	32.17	123.5m	14.1ms	17.8
OPLT$_{B,\alpha=0.625,p=3}$	54.01	34.90	26.04	298.7m	6.1ms	18.0	40.06	35.79	32.65	118.9m	14.0ms	16.0
OPLT$_{B,\alpha=0.75,p=3}$	54.69	35.32	26.31	297.0m	5.8ms	16.0	41.09	36.65	33.42	111.9m	13.0ms	16.0
OPLT$_{B,\alpha=0.875,p=3}$	54.56	35.22	26.25	285.4m	5.8ms	15.0	41.18	36.75	33.53	110.1m	12.7ms	15.0

Table 8: Mean precision at $\{1, 3, 5\}$ (%, averaged over 5 runs) of IPLT trained only on warm-start training dataset and IPLT with a tree created on warm-start training dataset (IPLT-U) and updated with all examples without updating the tree and OPLT with warm-start (OPLT-W). Notation: P@k – precision at k-position, w – percentage of the training dataset sampled for warm-start, p – number of passes over train dataset.

	AmazonCat			Wiki10			WikiLSHTC			Amazon		
Algorithm	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
IPLT$_{w=5\%,p=3}$	88.49	70.48	55.49	80.69	61.68	51.85	38.95	22.74	16.37	10.98	9.55	8.77
IPLT-U$_{w=5\%,p=3}$	93.15	78.54	63.62	83.95	72.99	63.20	54.70	34.54	25.27	29.54	22.65	18.22
OPLT-W$_{w=5\%,p=3}$	93.11	78.50	63.75	83.93	73.34	63.75	58.51	37.92	28.04	41.96	37.39	34.07
IPLT$_{w=10\%,p=3}$	89.92	72.97	57.99	80.79	65.46	55.38	44.59	26.71	19.35	15.77	13.70	12.45
IPLT-U$_{w=10\%,p=3}$	93.12	78.58	63.79	84.38	73.52	63.88	57.47	36.81	27.05	35.41	28.84	24.06
OPLT-W$_{w=10\%,p=3}$	93.10	78.51	63.79	84.75	74.02	64.33	59.23	38.39	28.38	42.21	37.60	34.25
IPLT$_{w=15\%,p=3}$	90.69	74.17	59.25	81.61	67.28	57.18	47.63	29.03	21.11	19.38	16.79	15.24
IPLT-U$_{w=15\%,p=3}$	93.13	78.62	63.85	84.65	73.93	64.24	58.55	37.71	27.77	37.90	31.87	27.19
OPLT-W$_{w=15\%,p=3}$	93.15	78.54	63.82	84.69	73.90	64.29	59.66	38.67	28.57	42.41	37.70	34.29