Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising

Yashoteja Prabhu*
yashoteja.prabhu@gmail.com

Anil Kag†
anilkagak2@gmail.com

Shrutendra Harsola†
shharsol@microsoft.com

Rahul Agrawal†
Rahul.Agrawal@microsoft.com

Manik Varma*†
manik@microsoft.com

ABSTRACT

This paper develops the Parabel algorithm for extreme multi-label learning where the objective is to learn classifiers that can annotate each data point with the most relevant subset of labels from an extremely large label set. State-of-the-art 1-vs-All approaches, which learn a separate classifier per label, have yielded significantly higher prediction accuracies as compared to leading tree and embedding based methods. Unfortunately, 1-vs-All approaches have training and prediction costs that are linear in the number of labels making them prohibitively expensive for real-world applications such as Dynamic Search Advertising (DSA). Parabel addresses this limitation by: (a) efficiently learning a balanced label hierarchy from training data; (b) generalizing the popular hierarchical softmax model to the multi-label setting so as to obtain a probabilistic model of the joint label distribution given the learnt hierarchy and (c) developing logarithmic time training and prediction algorithms based on the proposed model. This allows Parabel to be up to 600-900x faster at training and up to 60-13,000x faster at prediction as compared to leading 1-vs-All approaches while maintaining classification accuracy. Experiments also revealed that Parabel could efficiently scale to DSA problems involving 7 million labels where it significantly increased the ad-recall and clicks when added to the system in production on the Bing search engine. Parabel's source code can be downloaded from [1].

KEYWORDS

Extreme classification, dynamic search advertising, multi-label hierarchical softmax

ACM Reference Format:
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 *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3185998

*Indian Institute of Technology Delhi
†Microsoft Research & AI

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3185998

1 INTRODUCTION

Objective: This paper on extreme classification develops the Parabel algorithm whose predictions are almost as accurate as the state-of-the-art DiSMEC [4] and PPDSparse [44] classifiers while being up to 600-900x faster at training and up to 60-13,000x faster at prediction. This allows Parabel to efficiently and accurately predict the subset of 7 million Bing queries that might lead to a click on a given ad-landing page for Dynamic Search Advertising (DSA).

Extreme Classification: Extreme multi-label learning addresses the problem of automatically annotating each data point with the most relevant *subset* of labels from an extremely large label set. For instance, there are more than a million labels (tags) on Wikipedia and one might wish to build an extreme multi-label classifier that tags a new article with the subset of most relevant Wikipedia labels. Note that multi-label learning is distinct from multi-class classification which aims to predict a single mutually exclusive label.

DSA: Advertisers are required to provide only a product description, or an ad-landing page, in DSA. The search engine automates everything else including generating the ad-copy, customizing the ad-title and ad-landing page to the search query, generating the bid-phrases, *etc.* This provides a number of benefits to the advertiser including eliminating time-consuming and expensive tasks, reducing the deployment time for new and updated campaigns, more accurate targeting of users, *etc.*

A central problem in DSA is to determine the subset of search engine queries that might lead to a click on the given ad-landing page. The predictions need to be highly accurate as it is not possible to either have them be manually verified by the advertiser [3] or to have them be automatically matched against the advertiser supplied bid-phrases [33]. Inaccurate predictions therefore decrease user satisfaction, search engine revenue and conversions for the advertiser. At the same time, the predictions need to be made in milliseconds per test point so as to handle the large, and constantly evolving, corpus of billions of ads. Furthermore, low training costs are critical as multiple prediction models need to be trained, and frequently re-trained, for various markets with different languages and ever changing query distributions.

Traditional approaches for predicting search engine queries from ad-landing pages are based on landing page summarization [12], translation and query language models [35, 46], keyword suggestion based on Wikipedia concepts [48] and matching queries to the ad title based on deep embeddings [19, 36] *etc.* This paper formulates the problem as an extreme multi-label learning task instead with each of the top 7 million monetizable queries on Bing being treated

as a separate label. Extreme classifiers are trained to take in a bag-of-words representation of the ad-landing page as input and predict the subset of relevant Bing queries as output.

1-vs-All approaches: DiSMEC [4] and PPDSparse [44] make significantly more accurate predictions than all other extreme classifiers. They train a separate linear classifier per label based on the 1-vs-All approach and predict the label if the corresponding classifier fires for a given test point. This leads to training and prediction costs which grow linearly with the number of labels. In particular, given a dataset with N training points, N' test points, L labels and $\hat{D}$-sparse features, the training and prediction costs are $O(NL\hat{D})$ and $O(N'L\hat{D})$ respectively. Such costs can be prohibitive for real-world applications, such as DSA, where training and prediction can take months on datasets having millions of labels, hundreds of millions of training points and billions of test points.

Parabel: Parabel reduces the training time from linear to logarithmic in the number of labels while still following the 1-vs-All approach and learning a separate linear classifier per label. It achieves this by reducing the number of points used for training each classifier from $O(N)$ as in existing 1-vs-All approaches to $O((N/L)\log L)$. Unfortunately, traditional dataset subsampling techniques, such as PPDSparse's primal-dual optimization algorithm or the popular negative sampling heuristic [29] where negative training points are subsampled uniformly at random, can lead to a significant loss in prediction accuracy when applied aggressively. Parabel subsamples data points based on the intuition that accuracy can be maintained while restricting each label's negative training examples to those points annotated with the most similar, or confusing, labels. Thus, a balanced label hierarchy is learnt efficiently based on a novel label representation such that similar labels end up together at the leaves. The negative examples used for training a label's classifier are then drawn from the other labels in the same leaf as the given label. Parabel's 1-vs-All classifiers are learnt as a MAP estimate of the joint probability distribution over labels conditioned on data point features and the learnt hierarchy. The proposed model generalizes the popular multi-class hierarchical softmax model [10, 17, 29, 31] used for learning word vector embeddings and language models to the multi-label setting. Furthermore, the model also allows Parabel to make predictions in logarithmic time which are optimal for evaluation metrics such as precision@r and nDCG@r [20]. As a result, Parabel can be up to 600-900x faster at training and 60-13,000x faster at prediction as compared to DiSMEC and PPDSparse with almost the same prediction accuracy.

Comparison to tree based methods: Parabel can also outperform state-of-the-art tree based extreme classifiers. In particular, Parabel was found to have up to 20x lower training time, 38x lower model size and 9% higher prediction accuracy as compared to PfastreXML [20]. It achieves this by: (a) learning a small ensemble of up to 3 trees with powerful (1-vs-All) leaf node classifiers rather than a large ensemble of 50 PfastreXML trees with weak (constant) leaf node classifiers; (b) by learning trees which partition labels at each internal node rather than data points and (c) by learning internal and leaf node classifiers based on a principled probabilistic model.

DSA Results: These properties make Parabel better suited to web-scale applications, such as DSA, than leading extreme classifiers. DiSMEC and PPDSparse were unable to scale to DSA datasets while PfastreXML was 5% less accurate than Parabel in terms of offline metrics and had higher operating costs. Furthermore, Including Parabel in the Bing DSA ensemble in production led to a significant increase in online metrics including a 75% increase in ad-recall and a 42% increase in clicks.

Contributions: This paper makes the following novel contributions: (a) it proposes a balanced hierarchy over labels which is both more accurate and more efficient to learn as compared to existing hierarchies; (b) it develops a probabilistic model of the joint label distribution conditioned on data point features and the learnt label hierarchy which generalizes the hierarchical softmax model to the multi-label setting; (c) it develops efficient logarithmic time training and prediction algorithms based on the proposed model and (d) it demonstrates that Parabel could significantly improve the quality of dynamic search advertising on the Bing search engine.

2 RELATED WORK

Extreme multi-label learning: Much progress has recently been made in developing extreme multi-label learning algorithms based on trees [3, 20, 21, 33, 34, 37], embeddings [7, 11, 13, 18, 24, 30, 38, 41, 43] and 1-vs-All approaches [4, 25, 32, 42, 44, 45]. Of these, 1-vs-All and tree based approaches are directly relevant to this paper.

1-vs-All approaches: 1-vs-All approaches such as DiSMEC [4], PD-Sparse [45], PPDSparse [44] and XML-CNN [25] have high prediction accuracies and low model sizes. Unfortunately, these approaches also have high training and prediction times as they learn a separate classifier per label. Training costs can be high as each classifier has to be trained over millions of negative data points which might not be relevant for that label. PPDSparse [44] addresses this issue to a limited extent by optimizing over a shortlist of negative training examples based on a primal and dual sparsity preserving algorithm. This allows PPDSparse to be up to 100x faster at training than the state-of-the-art DiSMEC on some datasets while matching its prediction accuracy. Despite this speed-up, PPDSparse is unable to scale to large problems due to the significant overhead of generating the shortlist at each iteration. The prediction times of DiSMEC, PD-Sparse, PPDSparse and XML-CNN are also too high to meet the latency and throughput requirements of real-world applications such as DSA. Some heuristics have been proposed to speed up the prediction times of 1-vs-All approaches based on trees [42] and label filters [32]. Unlike Parabel, such approaches are applied post hoc after the base 1-vs-All classifiers have been trained and can therefore lead to decreased prediction accuracies and increased training times.

Tree approaches: Tree approaches to extreme classification have the advantages of low training and prediction times but have high model sizes and poor prediction accuracies. The Achilles heel of state-of-the-art tree approaches [3, 20, 34, 37] is their use of weak constant classifiers in their leaf nodes. Leading approaches therefore learn a large ensemble to compensate for the poor prediction accuracy of any single tree. Parabel differs by learning powerful 1-vs-All classifiers in its leaf nodes thereby allowing it to get significantly higher prediction accuracies using an ensemble of 3 trees. Furthermore, unlike most extreme multi-label tree classifiers [3, 20, 34, 37], Parabel learns a hierarchy over labels rather than data points. While trees that partition labels have been studied

extensively in the multi-class literature [5, 14, 16], most of these formulations do not extend straightforwardly to the multi-label setting and have not been shown to scale to millions of classes. For multi-label classification, the Probabilistic Label Tree (PLT) [21] and the HOMER [39] approaches are perhaps the most closely related to Parabel. Parabel improves over PLT by learning the label hierarchy rather than using a random one, by modelling the joint label distribution rather than the marginals and by developing more efficient optimizers. As demonstrated in Section 4, Parabel could be 10x faster to train than PLT as well as up to 20% more accurate. Parabel also differs from HOMER as it uses a more accurate label representation, has a more efficient algorithm for learning the label hierarchy and has principled algorithms for training and prediction based on a well-defined probabilistic model.

Deep learning: Deep learning approaches [22, 25, 49] focusing on learning representations for extreme classification have also been explored in the literature. Learning features is orthogonal to the focus of this paper though it should be noted that Parabel could be used to replace the tree [22] or 1-vs-All [25] classifiers used in these approaches for even better results.

DSA: This paper focusses on the problem of determining the subset of search engine queries that might lead to a click on a given ad-landing page. Various approaches have been proposed for this task in the organic search literature including information retrieval based methods [23], probabilistic methods and topic models [40] and deep learning [19, 36]. Unfortunately, such techniques have been found to not work well for pithy ad-landing pages. Techniques have also been proposed specifically for sponsored search including those based on landing page summarization [12], translation and query language models [35, 46] and keyword suggestion based on Wikipedia concepts [48]. Some of these approaches suffer from low coverage while others might not be able to effectively leverage historical click data. As demonstrated in Section 4, Parabel complements such approaches and significantly improves various online metrics when included in the Bing DSA ensemble in production.

3 PARABEL

This section describes Parabel's overall architecture and key components including an algorithm for efficiently learning a balanced label hierarchy which helps in accurately identifying the most similar labels to a given label, a multi-label model of the joint label probabilities given the learnt hierarchy and logarithmic time training and prediction algorithms based on the proposed model.

3.1 Architecture

Parabel learns a small ensemble of up to 3 label trees. Each label tree is grown by recursively partitioning the labels into two balanced groups. Nodes become leaves and are not partitioned further when they contain less than M labels. The leaf nodes contain linear 1-vs-All classifiers, one for each label in the leaf, trained on only those examples having at least one leaf node label.

A test point with unknown labels needs to traverse the label tree during prediction. Each internal node therefore learns two linear classifiers indicating whether a test point should be passed down to the left or right child or both. A test point therefore traverses multiple paths through the label tree to reach multiple leaves. The 1-vs-All classifiers in these leaves are evaluated to determine the probability that the corresponding labels are relevant to the test point. Predictions are made by averaging these probabilities across the various trees in the ensemble.

3.2 Learning the Label Hierarchy

This subsection describes the algorithm used to partition the labels at an internal node. The algorithm is applied recursively starting at the root node containing all the labels. Node partitioning is terminated when a node contains fewer than M labels. Training of the leaf node classifiers is discussed in Subsection 3.4.

Label representation and similarity: A measure of label similarity is needed for partitioning the labels allocated to an internal node. Parabel represents a label by a unit vector in the direction of the mean of the training points containing the label. Given a set of N training points $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with D dimensional feature vectors $\mathbf{x}_i \in \mathbb{R}^D$ and L dimensional label vectors $\mathbf{y}_i \in \{0,1\}^L$, label l is represented as

$$\mathbf{v}_l = \mathbf{v}'_l / \|\mathbf{v}'_l\|_2 \quad \text{where} \quad \mathbf{v}'_l = \sum_{i=1}^{N} y_{il} \mathbf{x}_i \tag{1}$$

and the similarity between labels l and j can be determined as $\mathbf{v}_l^\top \mathbf{v}_j$. Note that the label features are needed just as an intermediate representation to construct the label tree and do not contribute to Parabel's final model size. Further note that the features for all the labels can be computed efficiently in $O(N\hat{L}\hat{D})$ time where $\hat{L} = O(\log L)$ and $\hat{D}$ are the average label and feature sparsities respectively. This allows label similarity to be determined orders of magnitude more efficiently as compared to alternatives proposed in the literature such as those based on the label confusion matrix [5].

The proposed representation is based on the intuition that two labels are similar if they are active in similar training points. In DSA, two queries (labels) are similar according to the proposed representation if they lead to clicks on similar ads (training points). For example, the similarity between queries "car tyres online" and "all weather car tyres" is 0.94 as they lead to clicks on similar ads whereas the similarity between "all weather car tyres" and "dominos pizza" is 0.02 as they are dissimilar. Parabel's similarity measure might also be seen as an efficient proxy for the label confusion matrix since two similar labels, with similar training points, are likely to have high inter-class confusion. Furthermore, Parabel's label representation might be better suited for dealing with tail labels which have very few training points. The estimated label confusion matrix might be unreliable for such labels due to classifier overfitting and limited validation data. On the other hand, Parabel's error in estimating the tail label's mean vector might be lower leading to a more robust representation. Parabel's label representation can also be more suitable than HOMER's representation [39]

$$\tilde{\mathbf{v}}_l = [y_{1l}, y_{2l}, \cdots, y_{Nl}] \tag{2}$$

This can be observed from the fact that Parabel's label similarity $\mathbf{v}_l^\top \mathbf{v}_j$ might be high for similar tail labels l and j which are active for similar, but not the same, training points whereas HOMER's label similarity $\tilde{\mathbf{v}}_l^\top \tilde{\mathbf{v}}_j$ would be 0. As a result, Parabel's representation leads to significantly higher prediction accuracies than HOMER's as demonstrated in Section 4.

Label partitioning: The labels present at an internal node are partitioned into 2 balanced groups by clustering them using the constrained spherical $k = 2$-means objective

$$\max_{\boldsymbol{\mu}_\pm \in \mathbb{R}^D, \boldsymbol{\alpha} \in \{-1,+1\}^L} \frac{1}{L} \sum_{l=1}^{L} \left(\frac{1+\alpha_l}{2} \boldsymbol{\mu}_+^\top \mathbf{v}_l + \frac{1-\alpha_l}{2} \boldsymbol{\mu}_-^\top \mathbf{v}_l \right) \quad (3)$$

$$\text{s. t.} \quad \|\boldsymbol{\mu}_\pm\|_2 = 1, \quad -1 \leq \sum_{l=1}^{L} \alpha_l \leq 1 \quad (4)$$

where it has been assumed without loss of generality that the node has L labels and $\alpha_l = +1$ indicates that label l has been assigned to the positive cluster with mean $\boldsymbol{\mu}_+$ and $\alpha_l = -1$ indicates that it has been assigned to the negative cluster with mean $\boldsymbol{\mu}_-$. The constraint on $\boldsymbol{\alpha}$ ensures that the sizes of the two clusters are at most one apart though this could be relaxed to a user tunable hyperparameter if the label structure was known to be imbalanced *a priori*.

The optimization problem in (3) is NP-hard [6]. Parabel therefore employs the following alternating maximization algorithm which converges to a local optimum. The algorithm is initialized by sampling $\boldsymbol{\mu}_\pm$ from $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_L\}$ uniformly at random without replacement. The following two steps are then repeated in each iteration of the algorithm until convergence. In the first step of each iteration, (3) is maximized while keeping $\boldsymbol{\mu}_\pm$ fixed. It is straightforward to show that the optimal solution is given by $\alpha_l^* = \text{sign}\left(\text{rank}\left((\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^\top \mathbf{v}_l\right) - \frac{L+1}{2}\right)$ with sign(0) being resolved to +1 or −1 depending on whether the label is closer to the positive or negative cluster respectively. In the second step, each α_l is fixed and (3) is maximized with respect to $\boldsymbol{\mu}_\pm$ to get $\boldsymbol{\mu}_\pm^* = \boldsymbol{\mu}_\pm' / \|\boldsymbol{\mu}_\pm'\|_2$ where $\boldsymbol{\mu}_\pm' = \sum_{l:\alpha_l = \pm 1} \mathbf{v}_l$. Convergence is reached when the $\boldsymbol{\alpha}$ assignments do not change from the previous iteration. In practice, however, the algorithm was terminated early when the objective function did not increase by more than 10^{-4} from one iteration to the next. The derivation of the update equations and the proof of convergence to a local optimum are given in the supplementary material. Note that the proposed algorithm turns out to be a more efficient version of the constrained k-means algorithm for general k [9] where $\boldsymbol{\alpha}^*$ can be obtained in closed form rather than by solving a linear program. The label partitioning algorithm is applied recursively starting at the root node till all the trees are fully grown. Note that distinct trees are learnt as different random initializations lead to different solutions of (3).

Comparison to other approaches: Parabel's label tree is learnt in time $O(\hat{D}L \log L)$. This is significantly more efficient than partitioning labels using graph cuts on the label confusion matrix [5] which would not lead to balanced trees and where the time taken to partition the root node itself would be $O(L^2)$. The approaches in [14, 16] introduce balance constraints while partitioning labels at a node but their algorithms involve fitting an SVM at each iteration and can therefore be much more expensive than Parabel's k-means. As a result, none of the algorithms in [5, 14, 16] have been shown to scale to large problems involving millions of labels and training points. Furthermore, Parabel's label tree can lead to significantly more accurate predictions as compared to other scalable approaches such as constructing random label trees as done in PLT or learning the hierarchy based on HOMER's representation. Finally, Parabel should not be confused with LPSR [42], which also uses k-means to learn its trees albeit by clustering data points rather than by partitioning labels.

3.3 A Hierarchical Probabilistic Model

Notation: Given a label tree, let $\mathcal{I} = \{1, \ldots, N_\mathcal{I}\}$ denote the set of $N_\mathcal{I}$ internal nodes and $\mathcal{L} = \{N_\mathcal{I} + 1, \ldots, N_\mathcal{I} + N_\mathcal{L}\}$ denote the set of $N_\mathcal{L}$ leaf nodes in the tree. Furthermore, let C_n be the set of child nodes of an internal node n. Note that, while each internal node has only 2 children in the label tree proposed in Subsection 3.2, Parabel's probabilistic model holds for k-ary trees and thus it is not explicitly assumed that $|C_n| = 2$. Finally, let $\mathbf{y}_n$ be a vector of binary random variables denoting whether the labels in the leaf node n are relevant to a given data point $\mathbf{x}$.

Discriminative model: Parabel models the joint probability that a set of labels $\mathbf{y} \in \{0, 1\}^L$ is relevant to a given data point $\mathbf{x}$. A label set can be sampled from such a distribution as follows. At each internal node, starting at the root, a probability distribution is sampled to determine which child nodes should be traversed. This procedure is applied recursively until a set of leaf nodes is reached. The set of relevant labels is then obtained by sampling from the label distributions in the leaves that were reached.

To be more precise, let the binary random variable z_n take the value 1 if node n was traversed and 0 otherwise and let $\mathbf{z}_{C_n}$ be the set of the indicator z variables of the children of node n. Furthermore, let $\mathbf{z}$ denote the set of all the z variables in the tree. Then, tree traversal at the internal node n happens by sampling from $\mathbb{P}(\mathbf{z}_{C_n} | z_n = 1, \mathbf{x})$. The set of relevant labels $\mathbf{y}$ is generated by sampling labels from each leaf node n that has been reached according to $\mathbb{P}(\mathbf{y}_n | z_n = 1, \mathbf{x})$. Thus, Parabel's probabilistic model is given by

$$\mathbb{P}(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z}} \mathbb{P}(\mathbf{y}|\mathbf{z}, \mathbf{x}) \mathbb{P}(\mathbf{z}|\mathbf{x}) \quad (5)$$

$$= \sum_{\mathbf{z} \in \mathcal{Z}_\mathbf{y}} \prod_{n \in \mathcal{L}: z_n = 1} \mathbb{P}(\mathbf{y}_n | z_n = 1, \mathbf{x}) \prod_{n \in \mathcal{I}: z_n = 1} \mathbb{P}(\mathbf{z}_{C_n} | z_n = 1, \mathbf{x}) \quad (6)$$

where $\mathcal{Z}_\mathbf{y}$ denotes the set of all the configurations of $\mathbf{z}$ which could have led to $\mathbf{y}$ being sampled. The model is based on the following assumptions and theorem.

Unvisited node assumption: This assumption formalizes the observation that the children of an unvisited internal node will never be traversed and that the labels in an unvisited leaf node will never be sampled. This implies that

$$\mathbb{P}(\mathbf{z}_{C_n} = \mathbf{0} | z_n = 0, \mathbf{x}) = 1 \quad \forall n \in \mathcal{I} \quad (7)$$

$$\mathbb{P}(\mathbf{y}_n = \mathbf{0} | z_n = 0, \mathbf{x}) = 1 \quad \forall n \in \mathcal{L} \quad (8)$$

The set of $\mathbf{z}$ values which obeys this assumption is denoted by $\mathcal{Z}_\mathbf{y}$.

Subtree independence assumptions: Parabel assumes that the probability distribution at a visited node n, whether internal or leaf, is sampled independently of all the nodes that are outside the subtree rooted at node n such that

$$\mathbb{P}(\mathbf{z}_{C_n} | z_n = 1, \mathbf{x}) = \mathbb{P}(\mathbf{z}_{C_n} | z_n = 1, \mathbf{x}, \bar{\mathbf{y}}_{\mathcal{S}_n}, \bar{\mathbf{z}}_{\mathcal{S}_n}) \quad \forall n \in \mathcal{I} \quad (9)$$

$$\mathbb{P}(\mathbf{y}_n | z_n = 1, \mathbf{x}) = \mathbb{P}(\mathbf{y}_n | z_n = 1, \mathbf{x}, \bar{\mathbf{y}}_{\mathcal{S}_n}, \bar{\mathbf{z}}_{\mathcal{S}_n}) \quad \forall n \in \mathcal{L} \quad (10)$$

where $\bar{\mathbf{y}}_{\mathcal{S}_n}$ and $\bar{\mathbf{z}}_{\mathcal{S}_n}$ denote the sets of all the labels and all the z variables that lie outside the subtree rooted at node n. Note that these assumptions do not imply label independence. The leaf node assumption (10) allows Parabel to model arbitrary correlations

between the labels within a leaf while also modeling weaker correlations between label sets across leaves. In particular, labels in a leaf node n can have any correlation structure as long as they can be sampled efficiently from $\mathbb{P}(\mathbf{y}_n|z_n = 1, \mathbf{x})$. This allows Parabel to model the labels "car tyres online" and "all weather car tyres" as dependent if required. The assumption also does not imply that labels in different leaves are independent – they are only conditionally independent given $\mathbf{z}$. This allows labels such as "all weather car tyres" and "dominos pizza" to be modeled as being mutually exclusive.

The internal node assumption (9) implies that the decision about which child nodes to traverse at a visited internal node is taken independently of the decisions taken at all other nodes which are not its descendants. This is a natural assumption which places only mild restrictions on label correlations and is commonly made in most tree algorithms.

Theorem 3.1. **Tree factorization**: *Given a label tree, if the subtree independence and unvisited node assumptions hold at all the tree nodes, then for a label vector* $\mathbf{y}$ *and an indicator vector* $\mathbf{z} \in \mathcal{Z}_\mathbf{y}$

$$\mathbb{P}(\mathbf{y}|\mathbf{z}, \mathbf{x}) = \prod_{n \in \mathcal{L}: z_n = 1} \mathbb{P}(\mathbf{y}_n|z_n = 1, \mathbf{x}) \tag{11}$$

$$\mathbb{P}(\mathbf{z}|\mathbf{x}) = \prod_{n \in \mathcal{I}: z_n = 1} \mathbb{P}(\mathbf{z}_{C_n}|z_n = 1, \mathbf{x}) \tag{12}$$

Proof. Please see the supplementary material. □

Comparison with other models: The proposed multi-label model generalizes the multi-class hierarchical softmax model [10, 17, 29, 31] . Setting $k = M = 2$ in Parabel's model would learn a binary label tree where each internal node had two children and each leaf node contained a single label. Choosing all internal and leaf node distributions to be the logistic sigmoid [8] would then reduce Parabel's model to the multi-class hierarchical softmax as only a single, mutually exclusive label would be predicted. Parabel's model also differs from PLT [21] as it models the joint label distribution rather than the marginals over the individual labels. Nevertheless, Parabel can be reduced to PLT by setting each internal (leaf) node distribution to be the product of k (M) independent, binary logistic sigmoid distributions. This would allow each child node to be traversed independently of its siblings and allow each leaf node to predict all its labels independently. Various other instantiations of Parabel's model can be achieved based on different internal and leaf node distributions. Results for some of these models are presented in Section 4 and the supplementary material.

Tractability: Assume a balanced k-ary label tree over L labels grown such that each leaf node has fewer than M labels. The worst case space complexity of Parabel's probabilistic model is $O(L(2^k + 2^M))$ where it has been assumed that each leaf and internal node distribution takes $O(2^M)$ and $O(2^k)$ space respectively. This can be tractable for small values of M and k particularly as compared to the $O(2^L)$ space complexity needed for modeling any general multi-label distribution. Assuming that only $O(\log L)$ labels are active for a given data point, the cost of sampling these labels from Parabel's model is $O(2^M \log L + 2^k \log^2 L)$ which is also tractable for small M and k. The cost of evaluating $\mathbb{P}(\mathbf{y}|\mathbf{x})$ for a given $\mathbf{x}$ and $\mathbf{y}$ would depend on the cost of marginalizing over $\mathbf{z}$ and is discussed in Subsection 3.4.

3.4 Training

MAP estimation: Training Parabel involves learning the parameters of the internal and leaf node distributions $\mathbb{P}(\mathbf{z}|\mathbf{x})$ and $\mathbb{P}(\mathbf{y}|\mathbf{z}, \mathbf{x})$ respectively. Let $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_\mathcal{I}, \boldsymbol{\Theta}_\mathcal{L}\}$ denote Parabel's internal and leaf node distribution parameters and assume a training set of N independent and identically distributed points $\{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$ with $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0, 1\}^L$. Then, based on (6) and Theorem 3.1, the maximum *a posteriori* estimate of $\boldsymbol{\Theta}$ can be obtained as

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \mathbb{P}(\boldsymbol{\Theta}) \prod_{i=1}^N \mathbb{P}(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\Theta}) \tag{13}$$

$$= \arg\max_{\boldsymbol{\Theta}} \mathbb{P}(\boldsymbol{\Theta}) \prod_{i=1}^N \sum_{\mathbf{z}} \mathbb{P}(\mathbf{y}_i|\mathbf{z}, \mathbf{x}_i, \boldsymbol{\Theta}) \mathbb{P}(\mathbf{z}|\mathbf{x}_i, \boldsymbol{\Theta}) \tag{14}$$

$$= \arg\max_{\boldsymbol{\Theta}_\mathcal{I}, \boldsymbol{\Theta}_\mathcal{L}} \prod_{i=1}^N \sum_{\mathbf{z} \in \mathcal{Z}_{\mathbf{y}_i}} \Bigg(\prod_{n \in \mathcal{L}: z_n = 1} \mathbb{P}(\mathbf{y}_{in}|z_n = 1, \mathbf{x}_i, \boldsymbol{\Theta}_\mathcal{L}) \prod_{n \in \mathcal{I}: z_n = 1} \mathbb{P}(\mathbf{z}_{C_n}|z_n = 1, \mathbf{x}_i, \boldsymbol{\Theta}_\mathcal{I}) \Bigg) \mathbb{P}(\boldsymbol{\Theta}_\mathcal{I}) \, \mathbb{P}(\boldsymbol{\Theta}_\mathcal{L}) \tag{15}$$

where $\mathbf{y}_{in}$ is a vector such that $y_{inl} = 1$ if label l in node n is relevant to $\mathbf{x}_i$ and 0 otherwise, $z_n = 1$ if node n was traversed and 0 otherwise and $z_{C_n j} = 1$ if child j of node n was traversed and 0 otherwise.

Marginalization: Marginalization over $\mathbf{z}$ is intractable in the specific case when the leaf node distributions are given the freedom to not predict any labels whatsoever and the internal node distributions have the freedom to terminate traversal by not selecting any children. Evaluating $\mathbb{P}(\mathbf{y}|\mathbf{x})$ in this case would require marginalizing over the exponential configurations of $\mathbf{z} \in \mathcal{Z}_\mathbf{y}$ which could have led to $\mathbf{y}$ being sampled. One way of addressing this limitation would be to choose $\mathbb{P}(\mathbf{z}|\mathbf{x}, \boldsymbol{\Theta})$ such that tree traversal could not be terminated at an internal node and to choose $\mathbb{P}(\mathbf{y}|\mathbf{z}, \mathbf{x}, \boldsymbol{\Theta})$ such that each visited leaf predicted at least one label. The marginalization over $\mathbf{z}$ would then collapse as only a single $\mathbf{z}$ could have been used to sample the given $\mathbf{y}_i$. The supplementary material explores how such a tractable and accurate model could be obtained. An alternative solution, which would not preclude the choice of distributions assuming label independence, would be to allow arbitrary $\mathbb{P}(\mathbf{z}|\mathbf{x}, \boldsymbol{\Theta})$ and $\mathbb{P}(\mathbf{y}|\mathbf{z}, \mathbf{x}, \boldsymbol{\Theta})$ but to train Parabel by maximizing a lower bound approximation of the log likelihood. The lower bound could be obtained by choosing the sparsest $\mathbf{z}$ obtained as the union of all paths starting at the root and terminating at a leaf node containing at least one label present in $\mathbf{y}$. In either case, let $\mathbf{z} = \mathbf{z}_i$ represent the unique or the sparsest path traversed by $\mathbf{x}_i$ to generate $\mathbf{y}_i$. Then, the MAP estimation of (13) reduces to the following independent optimization problems over the internal and leaf nodes respectively

$$\min_{\boldsymbol{\Theta}_\mathcal{I}} -\log \mathbb{P}(\boldsymbol{\Theta}_\mathcal{I}) - \sum_{i=1}^N \sum_{n \in \mathcal{I}: z_{in} = 1} \log \mathbb{P}(\mathbf{z}_{iC_n}|z_{in} = 1, \mathbf{x}_i, \boldsymbol{\Theta}_\mathcal{I}) \tag{16}$$

$$\min_{\boldsymbol{\Theta}_\mathcal{L}} -\log \mathbb{P}(\boldsymbol{\Theta}_\mathcal{L}) - \sum_{i=1}^N \sum_{n \in \mathcal{L}: z_{in} = 1} \log \mathbb{P}(\mathbf{y}_{in}|z_{in} = 1, \mathbf{x}_i, \boldsymbol{\Theta}_\mathcal{L}) \tag{17}$$

which themselves decompose across the various internal and leaf nodes as the following independent problems

$$\min_{\mathbf{\Theta}_{\mathcal{I}n}} -\log \mathbb{P}(\mathbf{\Theta}_{\mathcal{I}n}) - \sum_{i:z_{in}=1} \log \mathbb{P}(\mathbf{z}_{iC_n}|z_{in}=1,\mathbf{x}_i,\mathbf{\Theta}_{\mathcal{I}n}) \quad (18)$$

$$\min_{\mathbf{\Theta}_{\mathcal{L}n}} -\log \mathbb{P}(\mathbf{\Theta}_{\mathcal{L}n}) - \sum_{i:z_{in}=1} \log \mathbb{P}(\mathbf{y}_{in}|z_{in}=1,\mathbf{x}_i,\mathbf{\Theta}_{\mathcal{L}n}) \quad (19)$$

where $\mathbf{\Theta}_{\mathcal{I}} = \{\mathbf{\Theta}_{\mathcal{I}1},\ldots,\mathbf{\Theta}_{\mathcal{I}N_{\mathcal{I}}}\}$ and $\mathbf{\Theta}_{\mathcal{L}} = \{\mathbf{\Theta}_{\mathcal{L}1},\ldots,\mathbf{\Theta}_{\mathcal{L}N_{\mathcal{L}}}\}$.

Convex optimization: Different choices of $\mathbb{P}(\mathbf{z}_{C_n}|z_n=1,\mathbf{x})$ and $\mathbb{P}(\mathbf{y}_n|z_n=1,\mathbf{x})$ lead to different trade-offs between training time, prediction time, model size and prediction accuracy – some of which are explored in the supplementary material. A balanced trade-off can be achieved by assuming that all nodes learn independent 1-vs-All classifiers such that $-\log \mathbb{P}(\mathbf{z}_{iC_n}|z_{in}=1,\mathbf{x}_i) = \sum_{j=1}^{k} \sigma_{inj}(z_{iC_nj})$ and $-\log \mathbb{P}(\mathbf{y}_{in}|z_{in}=1,\mathbf{x}_i) = \sum_{j=1}^{M} \sigma_{inj}(y_{inj})$ where $\sigma_{inj}(y) = \log(1+e^{-(2y-1)\mathbf{w}_{nj}^\top \mathbf{x}_i})$ denotes the log loss and where it has been assumed, without loss of generality, that leaf node n has M labels. Furthermore, applying l_2 regularization by assuming that $-\log \mathbb{P}(\mathbf{\Theta}_{\mathcal{I}n}) = \frac{1}{2C}\sum_{j=1}^{k}\|\mathbf{w}_{nj}\|^2$ and $-\log \mathbb{P}(\mathbf{\Theta}_{\mathcal{L}n}) = \frac{1}{2C}\sum_{j=1}^{M}\|\mathbf{w}_{nj}\|^2$ but then enforcing sparsity by clipping all the learnt weights less than a threshold ϵ to zero as suggested in DiSMEC [4] also helped balance model size and prediction time with training time and prediction accuracy. These assumptions lead to the following independent optimization problems for all classifiers j in all internal and leaf nodes n respectively

$$\min_{\mathbf{w}_{nj}} \frac{1}{2}\|\mathbf{w}_{nj}\|^2 + C\sum_{i:z_{in}=1} \sigma_{inj}(z_{iC_nj}) \quad (20)$$

$$\min_{\mathbf{w}_{nj}} \frac{1}{2}\|\mathbf{w}_{nj}\|^2 + C\sum_{i:z_{in}=1} \sigma_{inj}(y_{inj}) \quad (21)$$

Note that Parabel's overall optimization problem is convex as each of the problems in (20) and (21) is convex and can be optimized efficiently using Liblinear's [15] dual co-ordinate ascent algorithm. Furthermore, the independence of all the optimization problems can be exploited to make Parabel highly parallelized and distributed. This can make Parabel's training even faster than the single core implementation numbers reported in the experiments. Finally note that, while (20) and (21) lead to state-of-the-art results for low-dimensional, dense, deep learning features, even better results can be obtained for high-dimensional, sparse bag-of-words features by replacing the log loss by the squared hinge loss (though this does not correspond to a valid probability distribution).

Computational complexity: The first step in training Parabel is to compute the label representations at cost $O(N\hat{D}\log L)$. This assumes that $O(\log L)$ labels are relevant to a data point on average and where $\hat{D}$ is the average feature sparsity ($\hat{D}=D$ for dense features). A $k=2$-ary label tree is then learnt such that each leaf node contains at most $M=100$ labels (see Subsection 3.2). This has computational complexity $O(k\hat{D}L\log L)$ as there are $O(\log L)$ levels in the tree and clustering at each level has complexity $O(k\hat{D}L)$. Given the learnt label tree and a training point i, $\mathbf{z}_i$ is obtained as the union of all the paths connecting the root to a leaf containing any label relevant to point i. A point can therefore traverse multiple child nodes at any given internal node. Therefore, each child at every internal node trains an independent linear classifier to separate the training points reaching the child from the training points reaching its siblings according to (20). The complexity of learning the child node classifiers at each level is $O(kN\hat{D}\log L)$. The overall complexity of learning $O(\log L)$ levels of internal classifiers is therefore $O(kN\hat{D}\log^2 L)$. Finally, 1-vs-All classifiers are also trained in all the leaf nodes. Each classifier is trained to separate the training points annotated with a given label from the other training points reaching the label's leaf node according to (21). This implies that each classifier trains on $O(\frac{M}{L}N\log L)$ points on average. Therefore, the overall complexity of training all L leaf node classifiers is $O(NM\hat{D}\log L)$. Parabel's overall complexity for training T trees is therefore $O((L/N + M/k + \log L)TNk\hat{D}\log L)$. This compares favourably to leading 1-vs-All approaches, such as DiSMEC and XML-CNN, whose complexities are at least $O(NL\hat{D})$. PPDSparse tries to reduce training time by optimizing over a shortlist of negative training examples in each iteration of its algorithm. Unfortunately, the overheads of generating the shortlist make PPDSparse significantly slower than Parabel. Furthermore, PPDSparse's negative sampling heuristic is dependent on feature sparsity and is ineffective for dense features.

3.5 Prediction

Predictions need to be made efficiently in many extreme classification applications. This makes existing 1-vs-All approaches infeasible as their prediction times are linear in the number of labels. Parabel's prediction time is logarithmic in the number of labels and it can make accurate predictions in milliseconds on the largest datasets.

Optimal predictions for gain functions: Gain functions defined over the top ranked relevant predictions tend to be preferred for evaluating real-world ranking, recommendation and tagging applications as compared to traditional multi-label loss functions. Parabel's predictions therefore optimize such gain functions, including precision@r and nDCG@r, based on the following theorems.

Theorem 3.2. *Let $\mathbb{P}(\mathbf{y}|\mathbf{x})$ represent the joint probability that a set of labels $\mathbf{y}$ is relevant to point $\mathbf{x}$. Then, the ranking of labels according to their marginal probabilities as rank $\left(\{\mathbb{P}(y_l=1|\mathbf{x})\}_{l=1}^{L}\right)$ maximizes the expected gain of functions defined over the top ranked predictions alone such as precision@r and nDCG@r.*

Proof. Please see the supplementary material. □

Theorem 3.3. *Given a joint probability distribution $\mathbb{P}(\mathbf{y}|\mathbf{x})$ defined as in (6) over a label tree, the marginal probability of label l in leaf node n being relevant to point $\mathbf{x}$ is given by*

$$\mathbb{P}(y_l=1|\mathbf{x}) = \mathbb{P}(y_l=1|z_n=1,\mathbf{x}) \prod_{\hat{n}\in\mathcal{A}_n} \mathbb{P}(z_{\hat{n}}=1|z_{\mathcal{P}_{\hat{n}}}=1,\mathbf{x}) \quad (22)$$

where $\mathcal{A}_n$ is the set of ancestors of node n apart from the root and $\mathcal{P}_{\hat{n}}$ is the parent of $\hat{n}$.

Proof. Please see the supplementary material. □

Theorems 3.2 and 3.3 state that optimal precision@r and nDCG@r predictions can be obtained by sorting Parabel's marginal label probabilities and show how to compute the marginals. Computing $\mathbb{P}(y_l=1|z_n=1,\mathbf{x})$ and $\mathbb{P}(z_{\hat{n}}=1|z_{\mathcal{P}_{\hat{n}}}=1,\mathbf{x})$ in (22) would have

$O(2^M)$ and $O(2^k)$ cost respectively for arbitrary joint distributions $\mathbb{P}(\mathbf{y}_n|z_n = 1, \mathbf{x})$ and $\mathbb{P}(\mathbf{z}_{C_n}|z_n = 1, \mathbf{x})$. Thankfully though, both these probabilities can be obtained directly without any marginalization due to Parabel's independence assumptions in all leaf and internal nodes. In particular, $\mathbb{P}(y_l = 1|z_n = 1, \mathbf{x}) = 1/\left(1 + e^{-\mathbf{w}_{nl}^\top \mathbf{x}}\right)$ for label l in leaf node n and $\mathbb{P}(z_{\hat{n}} = 1|z_{\mathcal{P}_{\hat{n}}} = 1, \mathbf{x}) = 1/\left(1 + e^{-\mathbf{w}_{\mathcal{P}_{\hat{n}}\hat{n}}^\top \mathbf{x}}\right)$ for internal node $\mathcal{P}_{\hat{n}}$ with child node $\hat{n}$.

Beam search: Evaluating $\mathbb{P}(y_l|\mathbf{x})$ in (22) for all labels would incur $O(L)$ costs. Parabel avoids this by predicting only the top ranked relevant labels based on the following greedy, breadth-first tree traversal strategy. The strategy is based on the assumption that the most relevant labels for a given test point $\mathbf{x}$ can be determined by greedily traversing the P most probable paths at each level in Parabel's label tree. Starting at the root, each visited internal node in a level computes the probability of the paths from the root to each of its children according to (22). The P most probable paths are greedily selected to obtain the set of child nodes that should be traversed at the next level. Traversal terminates at P leaf nodes and has computational complexity $O(P\hat{D}k \log L)$. The marginal probabilities $\mathbb{P}(y_l = 1|\mathbf{x})$ of all the labels in each of the P visited leaf nodes is then calculated according to (22). This step has complexity $O(P\hat{D}M)$. Finally, Parabel predicts a score for each label l by averaging the label's marginal probability across all T trees in the ensemble as $s_l = \frac{1}{T}\sum_{t=1}^{T} \mathbb{P}_t(y_l = 1|\mathbf{x})$. Parabel's overall prediction complexity is therefore $O(TP\hat{D}k \log L + TP\hat{D}M)$ where $M \approx k \log L$. The value $P = 10$ was empirically found to work well for precision@5 and nDCG@5 as the top 5 labels were found to occur outside the top 10 paths with low probability.

4 EXPERIMENTS

Datasets: Experiments were carried out on datasets containing up to 12 million training points, 4 million dimensional features and 7 million labels (see Table 1 for dataset statistics). The applications considered include tagging Wikipedia articles (Wiki-500K [2] and WikiLSHTC-325K [7, 27]), item-to-item recommendation of Amazon products (Amazon-3M [2] and Amazon-670K [7, 27]), relevant query prediction for a given ad landing page (DSA-2M and DSA-7M) and document tagging (EURLex-4K [28]). All datasets, apart from DSA-2M and DSA-7M, can be downloaded from The Extreme Classification Repository [2]. The DSA datasets were created by mining the Bing logs. Each ad landing page was represented by a bag-of-words feature vector and the subset of 2M/7M queries that led to a click on the page became its relevant labels.

Algorithms and implementation: Parabel was compared to leading extreme classifiers including DiSMEC [4], PPDSparse [44], PD-Sparse [45] and XML-CNN [25] (1-vs-All based approaches), PfastreXML [20] and PLT [21] (tree based approaches) and LEML [47], WSABIE [41], CPLST [11], CS [18] and SLEEC [7] (embedding based approaches). All algorithms were trained on the bag-of-words feature representation provided on The Extreme Classification Repository apart from XML-CNN which is a deep learning method that learns its own features from the raw text directly.

Table 1: Dataset statistics

Dataset	Train N	Features D	Labels L	Test N'	Avg. labels per point	Avg. points per label
EURLex-4K	15,539	5,000	3,993	3,809	5.31	448.57
WikiLSHTC-325K	1,778,351	1,617,899	325,056	587,084	3.26	23.74
Wiki-500K	1,813,391	2,381,304	501,070	783,743	4.77	24.75
Amazon-670K	490,449	135,909	670,091	153,025	5.38	5.17
Amazon-3M	1,717,899	337,067	2,812,281	742,507	36.17	31.64
DSA-2M	11,966,195	4,091,864	2,078,535	2,988,996	2.57	14.82
DSA-7M	9,042,996	3,977,303	6,969,674	2,261,297	14.75	19.13

The implementations of all algorithms were provided by their authors apart from CPLST and CS. These algorithms were implemented by us while ensuring that the published results could be reproduced. Results have been reported for only those datasets to which an implementation scales. XML-CNN's results on WikiLSHTC-325K have also not been reported as the raw text was unavailable.

Parabel's implementation normalizes the features to unit l_2-norm so as to help deal with documents of different lengths. Results are reported for internal and leaf node classifiers trained using both the log loss and the squared hinge loss. All classifier weights less than $\epsilon = 0.5$ for log loss and $\epsilon = 0.1$ for squared hinge loss were clipped to 0 as recommended in DiSMEC [4]. The log loss was found to be better suited than the squared hinge loss for bag-of-words features evaluated using propensity-scored loss functions [20] and for deep learning features using unweighted losses (please see the supplementary material) while the squared hinge loss worked better for bag-of-words features evaluated using unweighted losses.

Hyperparameters: Parabel has 4 hyperparameters: (a) the number of trees (T); (b) the maximum number of paths that can be traversed in a tree (P); (c) the maximum number of labels in a leaf node (M) and (d) the misclassification penalty for all the internal and leaf node classifiers (C). The default parameter settings of $M = 100$, $P = 10$ and $C = 10$ ($C = 1$) for log loss (squared hinge loss) were used in all the experiments to eliminate hyperparameter sweeps (though tuning could have increased Parabel's accuracy). Results are reported for $T = 1 - 3$ trees on all datasets. The hyperparameters of the other algorithms were set as suggested by their authors wherever applicable and by fine grained validation otherwise.

Results on repository datasets: Table 2 compares Parabel's performance to leading 1-vs-All and tree classifiers using the popular precision@r gain function with $r = 1, 3$ and 5. Results for nDCG@r and propensity-scored variants [20] are reported in the supplementary material. All experiments were run on an Intel Xeon 2.5 GHz processor with 64 GB RAM except for XML-CNN training and prediction which were carried out on a Nvidia GTX TITAN X GPU. The training and prediction times for DiSMEC were estimated on training and test subsets as estimation on a single core was infeasible. Parabel could be up to 900x faster at training and 13,000x faster at prediction than DiSMEC while having prediction accuracies that were lower by 1% on the Amazon datasets and by 1.7% on Wiki-500K. PPDSparse and XML-CNN could not scale to Amazon-3M and were estimated to also be approximately 600x-1,000x slower to train than Parabel. On the smaller datasets, PPDSparse was found to be up to 4x slower at training and up to 60x slower at prediction than Parabel while having prediction accuracies that were 1.9% lower on WikiLSHTC-325K but 1.6% higher on Wiki-500K and 1% higher on Amazon-670K. Similarly, XML-CNN was up to 120x slower at training, 14x slower at prediction and had up to 9% lower prediction accuracies than Parabel. Finally, Parabel was also up to 8x faster to train, had up to 11x lower model size and had up to 9%

Table 2: Parabel is significantly faster at training and prediction than state-of-the-art extreme classifiers while having almost the same precision@$r = 1, 3, 5$ values. Results are reported for Parabel with $T = 1, 3$ trees trained using the log loss (l) and the squared hinge loss (s). XML-CNN times are not directly comparable as it was trained on a GPU. Please see the text for details.

Method	P1 (%)	P3 (%)	P5 (%)	Training time (hr)	Test time/point (ms)	Model size (GB)
EURLex-4K						
PfastreXML	75.45	62.70	52.51	0.087	3.92	0.41
PLT	76.58	62.99	52.16	1.30	8.64	0.012
CS	58.52	45.51	32.47	1.52	6.71	0.018
CPLST	72.28	58.16	47.73	2.20	6.82	0.018
WSABIE	68.55	55.11	45.12	0.20	0.39	0.018
LEML	63.40	50.35	41.28	0.64	3.53	0.035
SLEEC	79.26	64.30	52.33	0.062	7.57	0.13
XML-CNN	76.38	62.81	51.41	0.28	0.18	0.017
PD-Sparse	76.43	60.37	49.72	0.041	**0.12**	0.31
PPDSparse	**83.83**	**70.72**	**59.21**	0.015	1.14	0.065
DiSMEC	83.67	70.70	59.14	0.094	7.05	0.04
Parabel-l-T=3	81.91	68.50	57.54	0.063	1.01	0.038
Parabel-s-T=3	82.25	68.71	57.53	0.018	0.88	0.026
Parabel-s-T=1	81.52	67.83	56.49	**0.005**	0.28	**0.0086**
WikiLSHTC-325K						
PfastreXML	56.05	36.79	27.09	7.42	1.80	9.37
PLT	45.67	29.13	21.95	9.91	1.37	**0.52**
SLEEC	54.83	33.42	23.85	18.65	5.67	4.39
PD-Sparse	61.26	39.48	28.79	38.67	**0.17**	0.69
PPDSparse	64.08	41.26	30.12	3.93	37.76	5.14
DiSMEC	64.94	42.71	31.5	≈749	≈2622	3.79
Parabel-l-T=3	64.38	42.40	31.14	3.62	1.17	6.26
Parabel-s-T=3	**65.04**	**43.23**	**32.05**	1.03	1.27	3.10
Parabel-s-T=1	63.00	41.35	30.36	**0.29**	0.29	1.03
Amazon-3M						
PfastreXML	43.83	41.81	40.09	15.74	4.05	36.79
PPDSparse	-	-	-	≈3406	-	-
DiSMEC	**47.77**	**44.96**	**42.80**	≈4955	≈16430	39.71
Parabel-l-T=3	42.54	41.23	40.14	22.48	1.45	67.32
Parabel-s-T=3	47.51	44.68	42.58	5.39	1.20	31.43
Parabel-s-T=1	46.14	43.35	41.23	**1.73**	**0.32**	**10.47**
Wiki-500K						
PfastreXML	59.52	40.24	30.72	49.24	7.72	63.59
XML-CNN	59.85	39.28	29.81	117.23	23.20	3.71
PPDSparse	70.16	50.57	39.66	28.53	123.7	3.99
DiSMEC	**70.20**	**50.60**	**39.70**	≈7496	≈9355	14.76
Parabel-l-T=3	67.98	48.43	37.62	19.95	4.87	16.84
Parabel-s-T=3	68.52	49.42	38.55	8.18	3.27	5.69
Parabel-s-T=1	66.73	47.48	36.78	**2.76**	**0.97**	**1.90**
Amazon-670K						
PfastreXML	39.46	35.81	33.05	3.32	4.75	9.80
PLT	36.65	32.12	28.85	3.25	1.71	0.76
SLEEC	35.05	31.25	28.56	11.33	18.51	7.08
XML-CNN	35.39	31.93	29.32	52.23	16.18	1.49
PPDSparse	45.32	40.37	36.92	1.71	66.09	6.00
DiSMEC	**45.37**	**40.40**	**36.96**	≈373	≈1414	3.75
Parabel-l-T=3	43.90	39.42	36.09	1.23	1.38	5.33
Parabel-s-T=3	44.90	39.81	35.99	0.42	1.13	1.94
Parabel-s-T=1	43.29	38.03	34.07	**0.12**	**0.33**	**0.65**
DSA-2M						
PfastreXML	28.52	17.05	12.5	431.53	7.51	417.5
Parabel-l-T=3	32.07	18.64	13.52	61.21	5.20	53.75
Parabel-s-T=3	**33.44**	**20.21**	**14.79**	21.41	4.15	10.94
Parabel-s-T=1	31.26	18.83	13.74	**7.72**	**1.47**	**3.65**
DSA-7M						
PfastreXML	28.09	25.79	23.21	306.55	23.12	410.49
Parabel-l-T=3	31.95	29.42	26.40	104.39	11.24	227.69
Parabel-s-T=3	**32.84**	**30.28**	**27.35**	73.54	5.97	40.96
Parabel-s-T=1	30.77	28.35	25.61	**25.88**	**1.99**	**13.66**

Table 3: Alternative choices of Parabel's components leads to worse performance. Results have been reported in terms of precision@5. Please see the text for details.

Method	EURLex-4K	WikiLSHTC-325K	Amazon-670K
Rocchio leaf	36.71	22.40	30.74
Rocchio internal	50.06	22.44	27.53
Random tree	55.70	21.73	28.58
HOMER features	57.18	23.99	32.03
Parabel-s-T=3	**57.53**	**32.05**	**35.99**

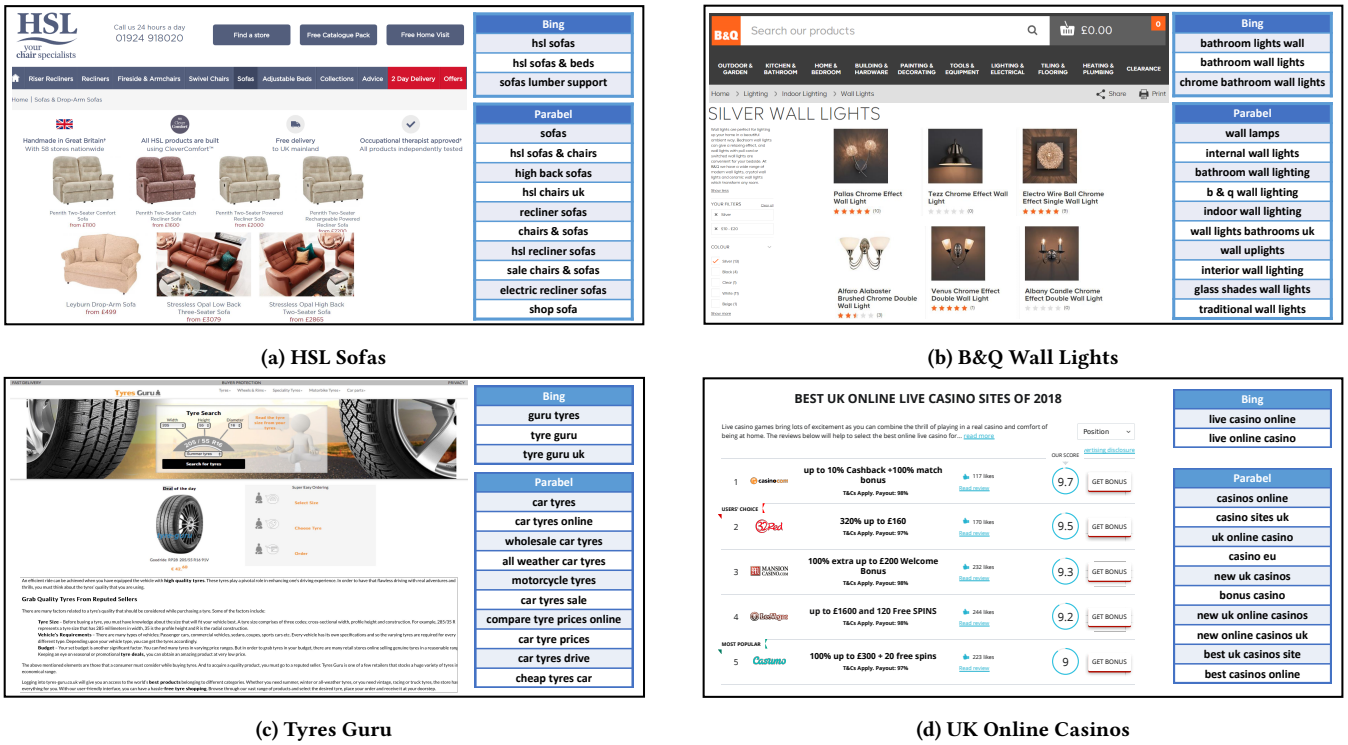
Table 4: The relative improvement of Parabel over BM25 on a live deployment of Dynamic Search Advertising on Bing.

Algorithm	Relative Ad-recall (%)	Relative CTR (%)	Relative BR (%)	Relative QOA (%)
Parabel	420	120	68	104

higher prediction accuracies than the leading tree based method PfastreXML [20].

Parabel variants: Table 3 demonstrates that Parabel's precision@5 decreases if its components were replaced by variants. Prediction accuracy could drop by up to 10% if Parabel's learnt label tree was replaced by PLT's random tree (Random tree). Furthermore, replacing Parabel's label representation by HOMER's label representation [39] (HOMER features) could reduce precision@5 by up to 8%. The training time could also be substantially reduced by replacing Parabel's strong 1-vs-All classifiers with weak Rocchio [26] classifiers in the internal (Rocchio internal) and leaf nodes (Rocchio leaf). However, accuracies could drop by up to 9% in either case. Finally, the results of varying Parabel's hyperparameters including the number of trees (T), the maximum number of labels in a leaf (M) and the number of paths taken (P) are reported in the supplementary material.

Dynamic Search Advertising: Table 2 also reports results on two small DSA datasets – DSA-2M having many more training points than labels and DSA-7M having almost the same number of training points and labels. None of the existing extreme classifiers could scale to these datasets apart from PfastreXML. Unfortunately, PfastreXML's performance was significantly worse than Parabel's. Parabel was 4 – 5% more accurate, was 4x-20x faster to train, had



(a) HSL Sofas

(b) B&Q Wall Lights

(c) Tyres Guru

(d) UK Online Casinos

Figure 1: Parabel can improve the quantity, quality and diversity of predicted queries from ad landing pages for DSA on Bing.

10x-38x lower model size and was 2x-4x faster at prediction. Parabel was therefore found to be better suited for a live deployment on Bing based on larger datasets.

Table 4 compares Parabel's performance to the BM25 information retrieval based algorithm [23] when deployed on Bing. BM25 ranks documents in response to a given query based on how frequently the query tokens occur in the document. The performance of both algorithms was evaluated based on the click-through rate (CTR), the bounce rate (BR) which is the percentage of times a user returned immediately to the search engine after clicking an ad, the ad-recall which is the percentage of ads that were clicked by at least one user and the quality of ad recommendations (QOA) which measures the goodness of ad recommendations according to a query-ad relevance model trained on human labelled data. As can be seen, Parabel improved relative ad-recall, CTR and BR by 320%, 20% and 32% respectively as compared to the BM25 strawman. Including Parabel in the ensemble of algorithms in production in Bing DSA generated 75% additional ad-recall and 42% additional clicks.

Figure 1 shows some qualitative examples where Parabel did better than the Bing ensemble in production. The traditional approaches in Bing were able to predict at most three queries for each ad landing page shown in Figure 1. Parabel predicted many more relevant queries thereby increasing the ad-recall, query-coverage and auction-density. This was particularly true for pithy ads. Parabel also increased the click count by making more diverse predictions which targeted specific aspects of each ad. For instance, Parabel was able to augment the "HSL Sofas" ad with queries such as "high back sofas", "recliner sofas", "electric recliner sofas", *etc.* Similarly, while Bing was able to only predict queries related to bathroom wall lights for the "B&Q Wall Lights" ad, Parabel also predicted "internal wall lights", "b&q wall lighting", "indoor wall lighting", *etc.* Finally, Parabel also increased the CTR and reduced the BR by avoiding predicting irrelevant queries.

5 CONCLUSIONS

This paper developed the Parabel algorithm for extreme multi-label learning. Parabel could make predictions almost as accurately as state-of-the-art extreme classifiers while having training and prediction costs that were logarithmic in the number of labels. Parabel's technical contributions include: a novel procedure for learning balanced label trees based on an efficient and informative label representation; a novel probabilistic hierarchical multi-label model which generalizes hierarchical softmax to the multi-label setting and scalable algorithms for efficient training and prediction. Experiments revealed that Parabel could be orders of magnitude faster at training and prediction as compared to leading 1-vs-All extreme classifiers with only a $1-2\%$ loss in accuracy in many cases. Parabel was also found to be superior to leading tree classifiers on all evaluation criteria. This made Parabel better suited for a live deployment on Bing where it significantly increased the ad-recall and clicks for dynamic search advertising.

ACKNOWLEDGEMENTS

We are grateful to Himanshu Jain, Kalina Jasinska, Krzysztof Dembczyński and Kunal Dahiya for helpful discussions and feedback.

REFERENCES

[1] [n. d.]. Code for Parabel. http://manikvarma.org/code/Parabel/download.html. ([n. d.]).

[2] [n. d.]. The Extreme Classification Repository. http://manikvarma.org/downloads/XC/XMLRepository.html. ([n. d.]).

[3] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. 2013. Multi-label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages. In *WWW*.

[4] R. Babbar and B. Schölkopf. 2017. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *WSDM*. 721–729.

[5] S. Bengio, J. Weston, and D. Grangier. 2010. Label Embedding Trees for Large Multi-class Tasks. In *NIPS*. 163–171.

[6] A. Bertoni, M. Goldwurm, J. Lin, and F. Saccà. 2012. Size Constrained Distance Clustering: Separation Properties and Some Complexity Results. 115 (2012), 125–139.

[7] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In *NIPS*.

[8] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

[9] P. S. Bradley, K. P. Bennett, and A. Demiriz. 2000. *Constrained K-Means Clustering*. Technical Report. MSR-TR-2000-65, Microsoft Research.

[10] W. Chen, D. Grangier, and M. Auli. 2015. Strategies for Training Large Vocabulary Neural Language Models. *CoRR* abs/1512.04906 (2015).

[11] Y. N. Chen and H. T. Lin. 2012. Feature-aware Label Space Dimension Reduction for Multi-label Classification. In *NIPS*.

[12] Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. R. Mediano, and B. Pang. [n. d.]. Using landing pages for sponsored search ad selection. In *WWW 2010*.

[13] M. Cissé, N. Usunier, T. Artières, and P. Gallinari. 2013. Robust Bloom Filters for Large MultiLabel Classification Tasks. In *NIPS*.

[14] J. Deng, S. Satheesh, A. C. Berg, and L. Fei-Fei. 2011. Fast and Balanced: Efficient Label Tree Learning for Large Scale Object Recognition. In *NIPS*. 567–575.

[15] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR* (2008).

[16] T. Gao and D. Koller. [n. d.]. Discriminative Learning of Relaxed Hierarchy for Large-scale Visual Recognition. In *ICCV*. 2072–2079.

[17] J. Goodman. 2001. Classes for Fast Maximum Entropy Training. 1 (2001), 561 – 564 vol.1.

[18] D. Hsu, S. Kakade, J. Langford, and T. Zhang. 2009. Multi-Label Prediction via Compressed Sensing. In *NIPS*.

[19] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

[20] H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications. In *KDD*.

[21] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hüllermeier. 2016. Extreme F-measure Maximization Using Sparse Probability Estimates. In *ICML*. 1435–1444.

[22] Y. Jernite, A. Choromanska, and D. Sontag. 2017. Simultaneous Learning of Trees and Representations for Extreme Classification and Density Estimation. In *ICML*.

[23] K. S. Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.* (2000).

[24] Z. Lin, G. Ding, M. Hu, and J. Wang. 2014. Multi-label Classification via Feature-aware Implicit Label Space Encoding. In *ICML*.

[25] J. Liu, W. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*. 115–124.

[26] C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[27] J. McAuley and J. Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*.

[28] E. L. Mencia and J. Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *SIGIR*.

[29] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*. 3111–3119.

[30] P. Mineiro and N. Karampatziakis. 2015. Fast Label Embeddings for Extremely Large Output Spaces. In *ECML*.

[31] F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. In *AISTATS*. 246–252.

[32] A. Niculescu-Mizil and E. Abbasnejad. 2017. Label ‘s for Large Scale Multilabel Classification. In *International Conference on Artificial Intelligence and Statistics*. 1448–1457.

[33] Y. Prabhu, A. Kag, S. Gopinath, K. Dahiya, S. Harsola, R. Agrawal, and M. Varma. 2018. Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation. In *WSDM*.

[34] Y. Prabhu and M. Varma. 2014. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*.

[35] S. Ravi, A. Z. Broder, E. Gabrilovich, V. Josifovski, S. Pandey, and B. Pang. [n. d.]. Automatic generation of bid phrases for online advertising. In *WSDM 2010*.

[36] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*.

[37] S. Si, H. Zhang, S. S. Keerthi, D. Mahajan, I. S. Dhillon, and C. J. Hsieh. 2017. Gradient Boosted Decision Trees for High Dimensional Sparse Output. In *ICML*. 3182–3190.

[38] Y. Tagami. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *KDD*. 455–464.

[39] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data*.

[40] X. Wei and W. B. Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR*.

[41] J. Weston, S. Bengio, and N. Usunier. 2011. Wsabie: Scaling Up To Large Vocabulary Image Annotation. In *IJCAI*.

[42] J. Weston, A. Makadia, and H. Yee. 2013. Label Partitioning For Sublinear Ranking. In *ICML*.

[43] C. Xu, D. Tao, and C. Xu. 2016. Robust Extreme Multi-label Learning. In *KDD*. 1275–1284.

[44] I. E. H. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing. 2017. PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *KDD*. 545–553.

[45] I. E. H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. S. Dhillon. 2016. PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*.

[46] W. T. Yih, J. Goodman, and V. R. Carvalho. [n. d.]. Finding advertising keywords on web pages. In *WWW 2006*.

[47] H. F. Yu, P. Jain, P. Kar, and I. S. Dhillon. 2014. Large-scale Multi-label Learning with Missing Labels. In *ICML*.

[48] W. Zhang, D. Wang, G. Xue, and H. Zha. 2012. Advertising Keywords Recommendation for Short-Text Web Pages Using Wikipedia. *ACM TIST* (2012).

[49] W. Zhang, L. Wang, J. Yan, X. Wang, and H. Zha. 2017. Deep Extreme Multi-label Learning. *CoRR* (2017).