Enabling Efficiency-Precision Trade-offs for Label Trees in Extreme Classification

Tavor Z. Baharav
Stanford University
tavorb@stanford.edu

Daniel L. Jiang*
University of Washington
danji@cs.washington.edu

Kedarnath Kolluri
Amazon
kkolluri@amazon.com

Sujay Sanghavi
Amazon
University of Texas at Austin
sujayrs@amazon.com

Inderjit S. Dhillon
Amazon
University of Texas at Austin
isd@amazon.com

ABSTRACT

Extreme multi-label classification (XMC) aims to learn a model that can tag data points with a subset of relevant labels from an extremely large label set. Real world e-commerce applications like personalized recommendations and product advertising can be formulated as XMC problems, where the objective is to predict for a user a small subset of items from a catalog of several million products. For such applications, a common approach is to organize these labels into a tree, enabling training and inference times that are logarithmic in the number of labels [23]. While training a model once a label tree is available is well studied, designing the structure of the tree is a difficult task that is not yet well understood, and can dramatically impact both model latency and statistical performance. Existing approaches to tree construction either optimize exclusively for statistical performance or optimize exclusively for latency. We propose an efficient information theory inspired algorithm to construct intermediate operating points that trade off between the benefits of both, which was not previously possible. We corroborate our theoretical analysis with numerical results, showing that on the Wiki-500K [4] benchmark dataset our method can reduce a proxy for expected latency by up to 28% while maintaining the same accuracy as Parabel [23]. On several datasets derived from e-commerce customer logs, our modified label tree is able to improve this expected latency metric by up to 20% while maintaining the same accuracy. Finally, we discuss challenges in realizing these latency improvements in deployed models.

CCS CONCEPTS

• **Mathematics of computing** → *Coding theory*; • **Information systems** → *Recommender systems*; • **Computing methodologies** → **Classification and regression trees**.

KEYWORDS

Extreme multi-label classification, Probabilistic label trees

*Work done while at Amazon.

CIKM '21, November 1–5, 2021, Virtual Event, Australia.

ACM ISBN 978-1-4503-8446-9/21/11.
https://doi.org/10.1145/3459637.3481914

ACM Reference Format:
Tavor Z. Baharav, Daniel L. Jiang, Kedarnath Kolluri, Sujay Sanghavi, and Inderjit S. Dhillon. 2021. Enabling Efficiency-Precision Trade-offs for Label Trees in Extreme Classification. In *Proceedings of the 30th ACM Int'l Conf. on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3481914

1 INTRODUCTION

With the ever increasing size of datasets, a new paradigm of classification problems has emerged in machine learning. In the setting of multi-label classification, the goal is to learn a model that can tag data points with a subset of relevant labels from a given label set. One common approach to this problem is 1-vs-All classification, where a separate classifier is learned for each label and all classifiers are evaluated at inference time, resulting in training and inference costs linear in the number of labels [4]. As the sizes of industrial datasets grow, the number of possible labels in these applications can easily reach hundreds of thousands or even millions, making the linear complexity of 1-vs-All methods prohibitive. This motivates the paradigm of extreme multi-label classification (XMC), where the number of labels, the number of points, and their dimensionality, are all extremely large [1]. Many modern large-scale industrial applications are routinely modeled as XMC problems, such as webpage annotation [21], text classification [12, 18, 30], dynamic search advertisement [23], and text similarity search [7].

To overcome the challenge of extremely large label spaces, many state of the art methods first organize the labels hierarchically into a search tree. These tree-based methods then learn a separate classifier for each internal node in the tree to predict whether a label relevant to the given context appears in the subtree rooted at that node. By utilizing the greedy traversal algorithm of beam search, these methods only evaluate the classifiers along a constant number of paths in the tree, resulting in efficient inference (with costs logarithmic in the number of labels for a balanced binary label tree) [23, 31]. Thus far, the design of this search tree has fallen into one of two categories: *similarity-based* [23] or *coding-theoretic* [12].

Similarity-based methods construct trees that optimize purely for statistical performance. These methods ensure that similar labels are placed close together in the tree. This helps yield meaningful label partitions at each internal node of the tree, and so the tree's internal classifiers can be expected to achieve high accuracy. An example of a model that structures its label tree this way is Parabel

[23], which constructs its tree by recursively applying balanced 2-means clustering to the set of label embeddings.

Such methods are commonly used in XMC solutions for e-commerce applications. One canonical example is product advertising, where the customer query is used as the input, and the catalog of sponsored products is the set of labels. Similarly, several forms of personalized recommendations can be modeled in the XMC framework. For example, the widget "Products related to this item" on an Amazon product page could be an XMC problem with the current product's information as the context and the products in Amazon's catalog as labels. In such applications, it is as important to retrieve results quickly as it is to have results that are relevant for the customer. Improvements in training or prediction efficiency of XMC models serving e-commerce applications can also yield savings in infrastructure costs, which must be considered alongside relevance.

Similarity-based trees, however, do not explicitly aim to address infrastructure costs. While tree-based methods scale the computational complexity from linear in the number of labels to logarithmic, further improvements have been achieved when some labels are more frequently matched to the context than others, causing an imbalance in label frequencies [18]. Not surprisingly, such an imbalance occurs in many XMC datasets, where the label frequencies are often well approximated by a power-law distribution [13]. For example, in Wiki-500K, a common benchmark XMC dataset, the most frequent 1% of the labels can provide at least one relevant label for 50% of the dataset (Figure 1, see Appendix A for more details). This phenomenon creates opportunity to further optimize training and inference costs by placing frequently occurring labels higher in the tree. Existing similarity-based methods do not perform this optimization, as they consider only the label feature space when clustering, placing every label at the same depth.

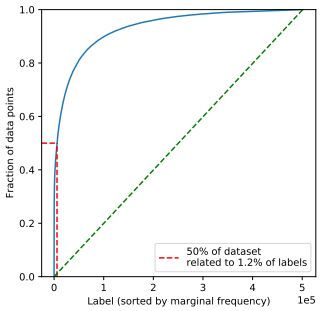


Figure 1: Label frequency imbalance on Wiki-500K.

On the other end of the spectrum, coding-theoretic trees optimize purely for the expected depth of the returned labels, ignoring the label feature space entirely. One model which utilizes such a tree is fastText [12], which applies Huffman coding to the label frequencies to construct a label tree for a hierarchical softmax. While such models yield efficient training and inference, observed speed-ups appear to be at the cost of accuracy [18]. For clarity and theoretical grounding, in this work we analyze the tree metric of expected depth (expected latency) as a proxy for training and prediction computational costs [18].

On the surface, these similarity-based and coding-theoretic methods are at an apparent impasse. The recent work of [6] on probabilistic label trees (PLTs), a formalization of the label trees previously discussed, posed a fundamental question that we tackle in this paper: *"to find a tree structure that results in a PLT with a low training and prediction computational costs as well as low statistical error seems to be a very challenging problem, not well-understood yet"*. In this paper, we design a scheme that can interpolate between similarity-based and coding-theoretic trees, allowing one to smoothly trade off between statistical performance and expected latency. Our contributions in this work are twofold:

(1) Provide a unified framework to study probabilistic label trees for datasets with both frequencies and similarity measures.
(2) Design an objective and algorithm for constructing PLTs with a tunable hyperparameter to interpolate between the computationally efficient and statistically efficient solutions.

Our solution trades off between label relevance and expected latency in a principled manner, and allows for operating points beyond simply these two extremes, as shown in Figure 2. While there is a general Pareto-style trade-off between these two metrics, we observe a surprising phenomenon across datasets; we are able to marginally *improve* statistical performance while reducing expected depth for some target operating points. On the Wiki-500K benchmark dataset, we see in Figure 2 that our method can reduce expected depth by 28% while maintaining the same accuracy. We also show that when we replace the coding-theoretic Huffman tree in fastText with a label tree constructed using methods described in this manuscript, we improve model accuracy on the Wiki-500K dataset by 40.6% while increasing expected depth by only 6.5%. We also show that incorporating modified label trees in Parabel improves the average depth traversed by up to 20% on several XMC datasets derived from e-commerce customer logs with no reduction in statistical performance.

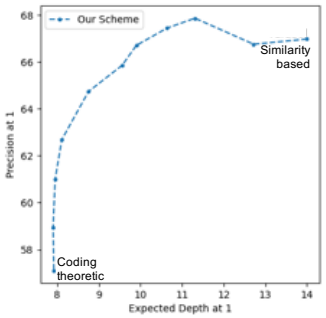


Figure 2: New operating points offered by our scheme, shown on Wiki-500K dataset.

We proceed by discussing related work and PLT background in Section 2. In Section 3 we discuss the two extremes of the spectrum that are present in the existing literature. In Section 4 we present our novel algorithm that interpolates between these two endpoints. In Section 5 we provide numerical results showing the improvement afforded by our scheme on e-commerce customer logs as well as public datasets, and provide theoretical backing to corroborate our experimental results. We conclude in Section 6.

2 RELATED WORK

XMC has seen a surge of work in recent years due to the rapidly increasing size of datasets. As previously discussed, one of the most accurate approaches to the XMC problem is to perform 1-vs-All classification [4]. 1-vs-All models such as DiSMEC [2], PD-Sparse [29], PPDSparse [28], and XML-CNN [15] learn a separate linear classifier for each label and evaluate all classifiers at inference time. While such methods often have good statistical performance, their inference times are necessarily linear in the number of labels.

Another class of approaches utilizes hashing to reduce an XMC problem down to a few small classification problems, saving storage by obviating the requirement of storing a tree. These methods have similar statistical performance to tree-based ones, but unfortunately require a fixed amount of time for each query, the same as a balanced tree, and cannot easily incorporate frequency information to improve expected latency [9, 25, 26].

2.1 Probabilistic Label Trees

With the increasing size of datasets, a linear dependency on the number of labels is no longer feasible in many e-commerce applications. A tree-based solution was first proposed in [20], which sped up the training time for a softmax by organizing the labels into a tree, creating a hierarchical softmax. This tree, with labels as its leaves, operates by routing inputs down the tree to a predicted relevant label (or more generally a set of labels). In this setting each internal node of this tree contains two classifiers, which estimate the probability that a data point has a relevant label contained in the left subtree or the right subtree. To ensure that the classifiers perform well, it is desirable to have a tree that obeys the similarity structure of the data; that is, labels that are similar to each other should be close together in this tree, whereas dissimilar labels should be far from each other. In the initial work of [20], a WordNet based clustering was hand-designed. However, it was later realized that these trees could be learned from data [19]. For this NLP task, the label embeddings for the softmax were hierarchically clustered into a tree via a divisive algorithm, where at every step the set of leaves is partitioned in half by fitting two Gaussians to their embeddings. Several recent works have focused on learning improved clusterings for higher accuracy trees [11, 13, 22, 24, 31]. Of particular interest to this work and e-commerce applications is Parabel [23], which recursively uses balanced spherical 2-means clustering to create a balanced binary tree on the labels. More generally, these label trees can be analyzed in the context of *Probabilistic Label Trees* (PLTs). We direct the interested reader to [6] for a formalization of PLTs and additional background.

2.2 Prefix-free coding

One shortcoming of existing PLT construction methods is that they are optimizing solely for the statistical accuracy of the tree; they do not optimize the inference time latency beyond that of a balanced binary tree. Starting in [20] with the hierarchical softmax, it was observed that this PLT construction could be viewed through the lens of prefix-free coding, as each path to a label in the tree can be seen as a binary string comprising the codeword for that given label. Information theory, specifically source coding, deals with the problem of representing a set of items in a minimum redundancy manner. Dating back to [8] researchers have worked on constructing optimal prefix-free codes; that is, constructing (binary) trees on a set of items such that the expected depth of an item selected randomly from a known distribution over these items would be minimized. The most well known such method was introduced by Huffman [10], which proceeds agglomeratively by iteratively merging the two least frequent items.

Working on hierarchical softmax, [17, 18] observed that Huffman codes could be utilized to construct PLTs. While similarity-based PLTs have good statistical performance, they have suboptimal latency because for common inputs one needs to traverse the entire depth of the tree every time, which is expensive. They noted that more frequent words should be higher in this tree to minimize expected compute time, and used Huffman coding to construct the PLT. This guaranteed optimal expected depth, minimizing training time. While faster, this approach yields worse statistical performance than using a word embedding-based label tree for the hierarchical softmax [19]. Subsequent work like fastText [12] also used a Huffman code to construct the PLT.

2.3 Best of both worlds

While statistical efficiency and computational efficiency are both desirable features on their own, in practice we want a solution that performs well in both of these metrics. Recently, the idea of combining these two approaches was considered in [27]. In their work, a Huffman code is generated on the words, and the output tree is post processed by rearranging the leaf nodes within a level to optimize for similarity of the word embeddings. This approach is often suboptimal however, as it first constructs a coding-theoretic PLT and then performs slight modifications to improve statistical performance, as opposed to optimizing for the two objectives simultaneously. One primary difficulty in this arises from the divisive nature of balanced 2-means clustering and other similarity-based tree construction methods, as compared to the agglomerative bottom up nature of Huffman and Shannon-Fano coding [6].

A motivating observation is that while previously utilized codes like Huffman and Shannon-Fano are agglomeratively constructed, there exist prefix-free binary codes that can be divisively constructed. In particular, one of the first prefix-free binary codes had this property: Fano coding [8], the direct predecessor of Huffman coding. Fano coding proceeds by successively sorting the items by frequency, and then partitioning them into two sets with as close to equal frequencies as possible. While Huffman coding yields an optimal prefix-free binary code (when symbols are coded individually), Fano coding yields a tree with near optimal expected depth, at most 1 worse than optimal [14]. However, due to its divisively constructed nature, it is much easier to merge with existing similarity-based tree construction techniques. In this work we take a step towards understanding this trade-off between statistical error and computational costs, and provide a scheme for constructing a PLT that interpolates between the two extremes.

3 TWO EXTREMAL TREES

To better describe our interpolating algorithm, we begin by describing in more detail its two endpoints: the similarity-based construction of recursive balanced 2-means clustering, and the coding-theoretic trees from Fano coding. Defining mathematical notation, in this problem we are given N data points $\boldsymbol{x_1}, \ldots, \boldsymbol{x_N} \in \mathbb{R}^d$, with corresponding label vectors $\boldsymbol{y_i} \in \{0,1\}^L$ with $y_{i,\ell}$ indicating whether label ℓ is relevant to $\boldsymbol{x_i}$. We use boldface to denote vectors, and define $\mathbf{1}$ as the all ones vector of appropriate dimension. In our binary clustering algorithms, we use the cluster assignment vector $\boldsymbol{\alpha} \in \{-1,+1\}^L$ to denote the assignment of label ℓ to cluster α_ℓ, where the left child contains all labels $\{\ell : \alpha_\ell = +1\}$ and the right child contains all labels $\{\ell : \alpha_\ell = -1\}$. The cluster centers we optimize over are $\boldsymbol{\mu_+}, \boldsymbol{\mu_-} \in \mathbb{R}^d$. Utilizing the label matrix $Y \in \{0,1\}^{N \times L}$, we construct the vector of marginal label frequencies $\boldsymbol{f} \propto Y^\top \mathbf{1}$. We now describe the two extremal schemes.

3.1 Balanced spherical 2-means clustering

Many existing similarity-based XMC tree models use some variant of balanced 2-means clustering to construct trees for their models. These methods cluster the labels utilizing high dimensional label

embeddings, which are constructed such that similar labels have similar label embeddings. In this work, we utilize Positive Instance Feature Aggregation (PIFA) embeddings $\boldsymbol{v_\ell} \in \mathbb{R}^d$ for label ℓ, which are constructed for a given label by averaging the training points that are relevant for that given label (further discussion in [23]).

Balanced spherical 2-means clustering is a common variant of k-means clustering for $k = 2$ using cosine similarity. In this setting the objects to be clustered, our L label embeddings $\{\boldsymbol{v_\ell}\}_{\ell=1}^L$, are all rescaled to have unit ℓ_2 norm, and our cluster centers $\boldsymbol{\mu_\pm}$ are restricted similarly. The objective is to find a cluster assignment vector $\boldsymbol{\alpha}$ and corresponding $\boldsymbol{\mu_\pm}$ that maximize the sum of similarities between $\boldsymbol{v_\ell}$ and it's corresponding cluster center ($\boldsymbol{\mu_+}$ if $\alpha_\ell = +1$, $\boldsymbol{\mu_-}$ if $\alpha_\ell = -1$). An additional balance constraint is enforced by requiring that $|\boldsymbol{\alpha}^\top \mathbf{1}| \leq 1$, restricting the two clusters to be of equal size. Mathematically, this optimization problem of balanced spherical 2-means clustering can be formulated as below

$$\max_{\substack{\|\boldsymbol{\mu_\pm}\|_2=1 \\ \boldsymbol{\alpha}\in\{-1,+1\}^L \\ |\boldsymbol{\alpha}^\top \mathbf{1}|\leq 1}} \sum_{\ell=1}^{L} \frac{1}{L}\left(\frac{1+\alpha_\ell}{2}\boldsymbol{v_\ell}^\top \boldsymbol{\mu_+} + \frac{1-\alpha_\ell}{2}\boldsymbol{v_\ell}^\top \boldsymbol{\mu_-}\right) \tag{1}$$

where α_ℓ corresponds to the cluster assignment of label ℓ, and defines the partitioning of labels to the left and right children. Due to the combinatorial constraint of $\boldsymbol{\alpha} \in \{-1, +1\}^L$, combined with the balanced constraint of $|\boldsymbol{\alpha}^\top \mathbf{1}| \leq 1$, this is an NP-hard problem [3]. Fortunately, we can still generate a good approximate solution efficiently via alternating maximization, leading to convergence to a local maxima. For a fixed cluster assignment $\boldsymbol{\alpha}$, we have that $\boldsymbol{\mu_\pm}$ are optimized as being proportional to their respective cluster centers. Concretely, $\boldsymbol{\mu_+} \propto \sum_{\ell:\alpha_i=+1} \boldsymbol{v_\ell}$ with $\boldsymbol{\mu_-}$ optimized similarly. Optimizing $\boldsymbol{\alpha}$ for fixed $\boldsymbol{\mu_\pm}$ requires the following:

(1) Sort the labels according to $\boldsymbol{v_\ell}^\top(\boldsymbol{\mu_+} - \boldsymbol{\mu_-})$
(2) Assign the first half of the labels in this sorted order as $\alpha_\ell = +1$, and the latter half as $\alpha_\ell = -1$
(3) If L is odd, assign the middle label ($\ell = \lfloor L/2 \rfloor + 1$) as $\alpha_\ell = -1 + 2 \cdot \mathbb{1}\{\boldsymbol{v_\ell}^\top(\boldsymbol{\mu_+} - \boldsymbol{\mu_-}) > 0\}$

By performing alternating maximization until the objective value increase from iteration to iteration falls below a specified threshold, we are able to efficiently generate a high quality clustering $\boldsymbol{\alpha}$, and use this to partition the labels. Constructing a tree by recursively applying balanced spherical 2-means clustering will ensure that similar labels are close to each other in the tree, but will completely ignore the label frequencies and place all labels at the same depth.

3.2 Fano tree

We now examine the details of a Fano coding based tree [8], which is designed to improve computational efficiency by reducing the expected depth required to traverse at test time. This scheme proceeds divisively similarly to spherical 2-means clustering, but instead of splitting the labels into the two most similar groups, it splits them into two sets of equal frequency.

This is accomplished in Fano coding by sorting the labels by frequency, and iteratively adding the the largest remaining frequency to the left child until its total frequency surpasses 1/2, sending the remaining labels to the right child. Reformulating this into an optimization problem outright is difficult due to the combinatorial nature of this task, and so instead we relax the constraints and allow ourselves to fractionally allocate the middle label to give each child a frequency of exactly 1/2. In order to force higher frequency labels to the left child, having $\alpha_\ell = +1$, we assign value f_ℓ^2 to this choice, noting that any super-linear function of f_ℓ can be used. This leads to our reformulation of each recursive call of the Fano coding scheme as a linear program (LP):

$$\max_{\substack{\boldsymbol{\alpha}\in[-1,+1]^L \\ \boldsymbol{\alpha}^\top \boldsymbol{f}=0}} \sum_{\ell=1}^{L} \alpha_\ell f_\ell^2. \tag{2}$$

By the fundamental theorem of linear programming the objective attains its maxima on a corner point, and so (2) is maximized by letting $\alpha_\ell = +1$ for the largest frequency items and -1 for the smaller ones. Solving this LP may yield one fractional α_ℓ (assuming the f_ℓ are unique) due to the $\boldsymbol{\alpha}^\top \boldsymbol{f} = 0$ constraint, which when changed to +1 will yield a valid Fano code assignment. Constructing a tree by recursively applying this Fano coding scheme yields a tree that prioritizes placing frequent labels at shallow depths to optimize computational efficiency, but ignores the label embeddings.

One important practical consideration is how we should construct the frequency vector our algorithm utilizes to minimize expected depth. While at first glance one may simply want to use the marginal label frequencies (number of occurrences of the label in the dataset), this can yield poor performance. This is because we want to minimize the expected depth one needs to search to in order to find k relevant labels for a given context. Considering the simple case of $k = 1$, if we have two very frequent labels that always show up together we do not need to put both of them high up in the tree. We do not care about the label's marginal frequency; we only care about if it will prevent contexts from needing to search deeper to find k relevant labels. There are several possible ways to create such a frequency vector suited for minimizing expected depth, which we denote $\widetilde{\boldsymbol{f}}$. To preserve the flow of this work, we relegate the construction of $\widetilde{\boldsymbol{f}}$ and further discussion on it to Appendix B.

4 INTERPOLATED LABEL TREES

Balanced 2-means clustering and Fano coding are two seemingly disparate algorithms with the former operating solely on the label embeddings and the latter operating solely on the label frequencies. We would ideally have a scheme that utilizes both similarity and frequency information, with a tunable knob offering a range of operating points trading off between precision and efficiency so that the right model can be selected for the specific application at hand. In online inference for example, a high-efficiency solution may be desired, even at the expense of a slight drop in accuracy, while for batch inference tasks a high-accuracy solution may be more useful, due to the less stringent latency constraints. A priori, it is unclear how to construct any intermediary point, let alone to interpolate between the two. In this section, we show that we can achieve many desirable intermediary operating points, as shown visually in Figure 2.

4.1 Weighted 2-means clustering

We begin by constructing one such intermediary point, defining a new objective we call weighted 2-means clustering. This can best

be understood by considering a simple case where each frequency is some integer multiple of a common base frequency. Then, one potential clustering scheme is to duplicate each point proportional to its frequency, and run balanced 2-means clustering on this expanded set of points, where after each iteration we assign each original point to the cluster where the majority of its duplicates fall. Mathematically, our weighted 2-means objective can be formulated as an LP:

$$\max_{\substack{\|\boldsymbol{\mu}_{\pm}\|_2=1 \\ \boldsymbol{\alpha}\in[-1,+1]^L \\ \boldsymbol{\alpha}^\top \boldsymbol{f}=0}} \sum_{\ell=1}^{L} f_\ell \left(\frac{1+\alpha_\ell}{2} \boldsymbol{v_\ell}^\top \boldsymbol{\mu_+} + \frac{1-\alpha_\ell}{2} \boldsymbol{v_\ell}^\top \boldsymbol{\mu_-} \right). \tag{3}$$

This constitutes making the following two changes to the balanced 2-means LP in (1).

(1) Follow a frequency weighted 2-means objective; weight how well label ℓ is matched by its frequency f_ℓ instead of $\frac{1}{L}$.
(2) Change the constraint $|\boldsymbol{\alpha}^T \mathbf{1}| \leq 1$ to $\boldsymbol{\alpha}^T \boldsymbol{f} = 0$, and relax α_ℓ to be contained in $[-1, 1]$.

Examining our alternating minimization algorithm, we have that for a fixed cluster assignment $\boldsymbol{\alpha}$, $\boldsymbol{\mu_\pm}$ are now optimized as the *weighted* means of their clusters. For fixed $\boldsymbol{\mu_\pm}$, $\boldsymbol{\alpha}$ is optimized similarly to before. Whereas for balanced 2-means we sorted the indices by $(\boldsymbol{\mu_+} - \boldsymbol{\mu_-})^\top \boldsymbol{v_\ell}$ and assigned the smaller half to cluster 1, we now assign as many labels as we can, proceeding in decreasing order of $(\boldsymbol{\mu_+} - \boldsymbol{\mu_-})^\top \boldsymbol{v_\ell}$, until their frequencies sum to over 1/2, and fractionally divide the α_ℓ of this last label ℓ to achieve $\boldsymbol{\alpha}^\top \boldsymbol{f} = 0$.

We relax the combinatorial problem of optimizing over $\alpha_\ell \in \{-1, +1\}$ to the LP optimizing over $\alpha_\ell \in [-1, +1]$, as otherwise we cannot ensure this frequency balance constraint. If we relax the balance constraint to be $|\boldsymbol{f}^\top \boldsymbol{\alpha}| < c$ and maintain the constraint on our α_ℓ to be discrete, then even alternating maximization will be difficult. This is because for fixed $\boldsymbol{\mu_\pm}$, due to the unequal costs (frequencies) of labels, assigning the α_ℓ's becomes a knapsack problem.

This formulation neatly obeys our intuition on what a similarity and frequency based clustering should look like. It places higher frequency labels higher in the tree, while also placing similar labels near each other. Hence, this weighted 2-means clustering falls in between the heavily imbalanced Fano tree and the the fully balanced tree from balanced 2-means.

4.2 Interpolating with a combined objective

Equipped with this one intermediary point, we now develop a more fine grained control of the trade-off between frequency and similarity information. We formulate below a combined objective that interpolates between these three operating points using a user specified hyperparameter $\lambda \in [0, 2]$. We use standard mathematical notation with $(x)_+ \triangleq \max(x, 0)$ and $x \wedge y \triangleq \min(x, y)$.

$$\max_{\substack{\|\boldsymbol{\mu}_{\pm}\|_2=1 \\ \boldsymbol{\alpha}\in[-1,+1]^L \\ \boldsymbol{\alpha}^\top \boldsymbol{f}(\lambda)=0}} \sum_{\ell=1}^{L} f_\ell(\lambda)(2-\lambda) \left(\frac{1+\alpha_\ell}{2} \boldsymbol{v_\ell}^\top \boldsymbol{\mu_+} + \frac{1-\alpha_\ell}{2} \boldsymbol{v_\ell}^\top \boldsymbol{\mu_-} \right) + \alpha_\ell (\lambda-1)_+ f_\ell(\lambda)^2 \tag{4}$$

$$\text{where } f_\ell(\lambda) = \frac{(2-\lambda) f_\ell^{\lambda \wedge 1} + (\lambda-1)_+ \widetilde{f}_\ell + \gamma/L}{(2-\lambda) \sum_{j=1}^{L} f_j^{\lambda \wedge 1} + (\lambda-1)_+ + \gamma}$$

To understand the above LP, first consider setting the additive smoothing parameter $\gamma = 0$. Then, this LP yields balanced 2-means clustering for $\lambda = 0$, our intermediary weighted 2-means clustering at $\lambda = 1$, and Fano coding at $\lambda = 2$. It smoothly interpolates between these three points, and for $\lambda \in (0, 1)$ can be seen as applying a shrinkage function on the frequencies with $f_\ell(\lambda) \propto f_\ell^\lambda + \gamma/L$. The interpolation is achieved by linearly placing less weight on the 2-means objective as λ ranges from 0 to 2, and gradually changing the frequency vector $\boldsymbol{f}(\lambda)$ from constant at $\lambda = 0$ to $\boldsymbol{f}(\lambda) \propto \boldsymbol{f} + \gamma/L$ when $\lambda = 1$ (balanced 2-means to weighted 2-means). $\widetilde{\boldsymbol{f}}$ plays no role when $\lambda \in [0, 1]$, as the nonlinear $(\lambda - 1)_+$ is only active when $\lambda \in (1, 2]$. In this regime, the frequency vector linearly trades off between $\boldsymbol{f}$ and $\widetilde{\boldsymbol{f}}$, as our objective is $(2 - \lambda)$ times the weighted 2-means objective plus $(\lambda - 1)_+$ times the Fano objective on $f_\ell(\lambda)$. The additive Laplacian smoothing parameter $\gamma > 0$ is to guard against distributional mismatch between train and test sets.

Constraining the coordinates of $\boldsymbol{\alpha}$ to be in $\{-1, +1\}$ leads to a combinatorial optimization problem, with the inherent difficulties this brings (as mentioned before). Fortunately however, relaxing the constraints to $\boldsymbol{\alpha} \in [-1, +1]^L$ until rounding the final solution makes this objective easily optimizable via alternating maximization, stopping when the objective value no longer increases. Even though the optimization problem in (4) is nonconcave, the objective is bilinear in $\boldsymbol{\alpha}$ and $\boldsymbol{\mu_\pm}$, and so each phase of alternating maximization is optimizing a linear objective (possibly with a quadratic constraint), which we show can be done efficiently.

Optimizing $\boldsymbol{\mu_\pm}$: fixing $\boldsymbol{\alpha}$, and decoupling $\boldsymbol{\mu_+}$ and $\boldsymbol{\mu_-}$, we see that $\boldsymbol{\mu_+}$ is optimized as

$$\operatorname*{argmax}_{\|\boldsymbol{\mu_+}\|_2=1} \sum_{\ell=1}^{L} f_\ell(\lambda)(1+\alpha_\ell) \boldsymbol{v_\ell}^\top \boldsymbol{\mu_+} \propto \sum_{\ell=1}^{L} f_\ell(\lambda)(1+\alpha_\ell) \boldsymbol{v_\ell},$$

where $\boldsymbol{\mu_-}$ can be optimized similarly. This means that once given cluster assignments $\boldsymbol{\alpha}$, $\boldsymbol{\mu_\pm}$ are optimized as being proportional to their weighted cluster centers, requiring $O(Ld)$ time.

Optimizing $\boldsymbol{\alpha}$: fixing $\boldsymbol{\mu_\pm}$, the optimal $\boldsymbol{\alpha}$ can be solved for as

$$\begin{aligned} \alpha^\star &= \operatorname*{argmax}_{\substack{\boldsymbol{\alpha}\in[-1,+1]^L \\ \boldsymbol{\alpha}^\top \boldsymbol{f}(\lambda)=0}} \sum_{\ell=1}^{L} \alpha_\ell \underbrace{\left(f_\ell(\lambda) \frac{2-\lambda}{2} \boldsymbol{v_\ell}^\top (\boldsymbol{\mu_+} - \boldsymbol{\mu_-}) + (\lambda-1)_+ f_\ell(\lambda)^2 \right)}_{\beta_\ell} \\ &= \operatorname*{argmax}_{\substack{\boldsymbol{\alpha}\in[-1,+1]^L \\ \boldsymbol{\alpha}^\top \boldsymbol{f}(\lambda)=0}} \boldsymbol{\alpha}^\top \boldsymbol{\beta} \end{aligned} \tag{5}$$

Due to the nice structure of this LP, once we compute $\boldsymbol{\beta}$ which requires $O(Ld)$ time, we are able to solve the LP efficiently in $O(L)$ time, as stated in the following lemma.

Lemma 1. *An optimal $\boldsymbol{\alpha}^\star$ for (5) can be constructed as:*

(1) *Sort labels by $\beta_\ell / f_\ell(\lambda)$, initialize all $\alpha_\ell^\star = -1$*
(2) *Starting from the largest $\beta_\ell / f_\ell(\lambda)$, iteratively assign $\alpha_\ell^\star = +1$ until $\boldsymbol{\alpha}^{\star\top} \boldsymbol{f}(\lambda) > 0$*
(3) *Assign the last label ℓ that was set to $\alpha_\ell^\star = 1$ fractionally to achieve $\boldsymbol{\alpha}^{\star\top} \boldsymbol{f}(\lambda) = 0$*

Proof of Lemma 1. We show that this $\boldsymbol{\alpha}^\star$ is an optimal solution for (5) by analyzing the dual LP. To start, we modify (5) to obtain an LP in standard form by shifting the α_ℓ to range between $[0, 1]$

instead of $[-1, 1]$ (the original $\boldsymbol{\alpha}$ can be obtained by shifting and rescaling). We replace $\boldsymbol{f}(\lambda)$ by $\boldsymbol{f}$ for brevity, and denote the optimal primal value as $p^\star$, where

$$p^\star = \max_{\substack{0 \le \boldsymbol{\alpha} \le 1 \\ \boldsymbol{\alpha}^\top \boldsymbol{f} = 1/2}} \boldsymbol{\alpha}^\top \boldsymbol{\beta}. \tag{6}$$

We prove Lemma 1 by showing that there exists a feasible solution to the dual LP of (6), which achieves objective value equal to that of our $\boldsymbol{\alpha}^\star$ in the primal. We can construct such an optimal $\boldsymbol{\alpha}^\star$ by initializing all coordinates to 0, sorting the indices by β_ℓ / f_ℓ, iteratively assigning $\alpha^\star_\ell = 1$ until $\boldsymbol{\alpha}^{\star\top} \boldsymbol{f} > 1/2$, then setting this final adjusted index (which we call ℓ_t) to have $\alpha^\star_{\ell_t} = \frac{1}{f_{\ell_t}} \left(\frac{1}{2} - \sum_{i \in \mathcal{S}} f_i \right)$. Defining the set of indices where $\alpha^\star_i = 1$ from the above scheme as $\mathcal{S}$, we obtain $p^\star \ge \sum_{i \in \mathcal{S}} \beta_i + \alpha^\star_{\ell_t} \beta_{\ell_t}$. Denoting the $L \times L$ identity matrix by I_L and defining $A = \begin{bmatrix} I_L & f & -f \end{bmatrix}^\top, \boldsymbol{b} = \begin{bmatrix} \mathbf{1}_L^\top & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}^\top$ we obtain our dual LP with value $d^\star$

$$d^\star = \min_{\substack{\boldsymbol{y} \ge 0 \\ A^\top \boldsymbol{y} \ge \boldsymbol{\beta}}} \boldsymbol{y}^\top \boldsymbol{b}. \tag{7}$$

Denoting by $\boldsymbol{y}^\star$ our proposed optimal solution to eq. (7), we see by complementary slackness that we can set $y^\star_i = 0$ whenever $[A\boldsymbol{\alpha}^\star - \boldsymbol{\beta}]_i > 0$. Considering the case where there is a fractional $\alpha^\star_i$, that is $\sum_{i \in \mathcal{S}} f_i < \frac{1}{2}$ and so $\alpha_{\ell_t} \in (0, 1)$, we must have that $y^\star_i = 0$ for $i \in \mathcal{S}^c$ and for $i = \ell_t$. To construct the remaining entries of $\boldsymbol{y}$, we set $y^\star_{L+1} = \beta_{\ell_t} / f_{\ell_t}$ with $y_{L+2} = 0$, and $y^\star_i = \beta_i - f_i y^\star_{L+1}$ for $i \in \mathcal{S}$; this allows us to satisfy $A^\top \boldsymbol{y}^\star \ge \boldsymbol{\beta}$. We then see that our dual objective value is

$$d^\star \le \boldsymbol{b}^\top \boldsymbol{y}^\star = \left(\sum_{i \in \mathcal{S}} \beta_i - f_i y^\star_{L+1} \right) + \frac{1}{2} y^\star_{L+1} = \sum_{i \in \mathcal{S}} \beta_i + \alpha^\star_{\ell_t} \beta_{\ell_t}. \tag{8}$$

Since the optimal dual objective value upper bounds the optimal primal objective value (i.e., $d^\star \ge p^\star$) and our $\boldsymbol{\alpha}^\star$ achieves the upper bound given by the dual in (8), $\boldsymbol{\alpha}^\star$ is an optimal solution for the primal [5]. Where this proof assumed that $\alpha_{\ell_t} \in (0, 1)$, the case of $\alpha_{\ell_t} = 0$ follows identically, with $\alpha_{\ell_t} = 1$ requiring us to increment the index ℓ_t to the next element in sorted order of β_ℓ / f_ℓ. □

Now that we have shown that each step of alternating maximization can be performed efficiently, we prove convergence of the overall procedure in the following theorem.

Theorem 1. *Performing alternating maximization on $\boldsymbol{\alpha}, \boldsymbol{\mu}_\pm$ for the optimization problem in* (4) *converges to a stationary point.*

Proof. Defining one iteration as comprising both a $\boldsymbol{\mu}_\pm$ and an $\boldsymbol{\alpha}$ optimization phase, we see similarly to the proof of Theorem 2.2 in [23] that since the objective value must increase over each iteration, no configuration of $\boldsymbol{\alpha}$ will be repeated. This is because the $\boldsymbol{\mu}_\pm$ at the end of two iterations must be the same if the $\boldsymbol{\alpha}$ at the end of those two iterations are the same. However, this would mean that the objective value did not increase in this iteration, and so the procedure would have terminated. We observe that there are at most 2^{L-1} possible $\boldsymbol{\alpha}$ vectors using our iteration scheme, as the $\boldsymbol{\alpha}$ vectors we construct have at most 1 fractional entry, which (if feasible) is uniquely determined by the other $L-1$ entries due to the balancedness constraint of $\boldsymbol{\alpha}^\top \boldsymbol{f}(\lambda) = 0$. As every non-fractional entry will be ± 1, this gives 2^{L-1} total $\boldsymbol{\alpha}$ that can appear at the end of an iteration. Thus, the algorithm will converge to a stationary point in a finite number of iterations. □

This result can be thought of as convergence to a Nash Equilibrium in a nonconcave game, where two players, one controlling $\boldsymbol{\mu}_\pm$ and one controlling $\boldsymbol{\alpha}$, alternate best responding to the other's actions. Due to the nonconcave objective and constraint set our guarantees for convergence of alternating maximization are to a stationary point and not to a local (or global) maxima.

5 EXPERIMENTS

We demonstrate the effectiveness of our scheme by using it to augment two widely used tree-based XMC models. Where the similarity-based trees optimize for precision and coding theoretic trees optimize for expected depth, we show that by varying λ we can easily interpolate between these two extremes. We provide results on the public benchmark XMC datasets AmazonCat-13K [16], Amazon-670K [16], and Wiki-500K [4]. We also provide results on 5 large e-commerce datasets with up to 10 million labels. Table 1 lists the number of training points N, the dimension of the points d, the number of labels L, the number of test points N', and the average number of labels per training point for each public dataset. Table 2 lists the same statistics for each e-commerce dataset.

Dataset	N	d	L	N'	Avg labels/pt
AmazonCat-13K	1.2M	204K	13K	307K	5.04
Wiki-500K	1.8M	2.4M	501K	784K	4.77
Amazon-670K	490K	136K	670K	153K	5.45

Table 1: Public XMC dataset statistics.

We first use our algorithm to augment the similarity-based XMC method Parabel [23]. In particular, we replace Parabel's tree with trees constructed by our algorithm, improving their expected depth on public datasets. We then use our algorithm to augment the coding-theoretic tree used by fastText. In particular, we replace fastText's Huffman-based hierarchical softmax with our algorithm's PLT, improving the statistical performance of the model on public datasets. Finally, we demonstrate the effectiveness of the augmented Parabel model on large e-commerce datasets, where we show an improvement in expected depth of up to 20%.

To measure statistical performance we use precision at k, which is computed as the fraction of true positive labels out of the k labels predicted by a model, averaged over the test contexts. To quantify computational efficiency, we utilize expected depth at k as a proxy for expected prediction latency, defined as the average depth (over the test contexts) searched to in our PLT. For a given context, this is obtained by looking at the top k predicted labels, and taking the depth of the deepest returned label. Creating a PLT prediction algorithm that realizes these expected depth gains as wall-clock improvements is left as future work, as this is an application and implementation-specific task. Additional experiment details can be found in Section 5.4.

5.1 Augmented Parabel

We augment Parabel with our trees for different values of λ and evaluate the performance on the large-scale public XMC datasets Amazon-670K and Wiki-500K in Figure 3. We interpolate the full

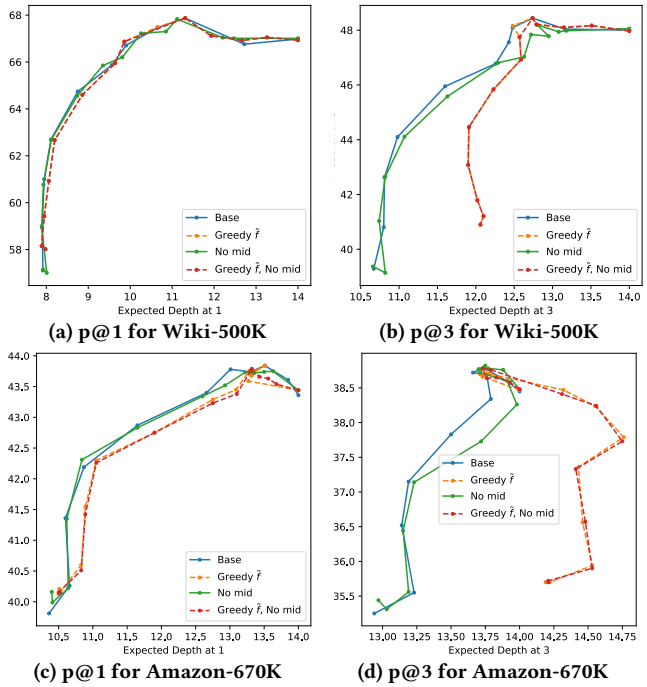
spectrum from fully similarity-based ($\lambda = 0$, standard Parabel) to coding-theoretic ($\lambda = 2$).

While there is a general Pareto-style trade-off between expected depth and precision, we observe a surprising phenomenon in our numerical experiments; we are able to marginally *improve* precision while reducing expected depth for small $\lambda > 0$. This means that, even without translating these expected depth gains into wall-clock improvements, we are able to improve model precision at effectively no cost. One possible explanation for this improvement is the fact that our interpolated scheme, compared to Parabel's clustering algorithm, reduces the depth of frequently accessed "popular" labels. Since each level in the tree compounds the error in routing and prediction, having frequent labels higher in the tree improves their prediction accuracy.

Additionally, $\widetilde{f}$ can be constructed via several different methods, as previously discussed. In Figure 3 we show several minor modifications to our original scheme (depicted in blue). In the orange curve, we construct $\widetilde{f}$ in a greedy manner; that is, we iteratively find the highest frequency label in the dataset, remove it and all contexts that contain it from the dataset (assigning it frequency equal to the number of removed contexts), and repeat. In the blue curve, we instead construct $\widetilde{f}$ by sorting the labels by their marginal frequencies, then iterate over each context and assign its frequency to the label it contains with the highest marginal frequency.

Another question one can ask is whether this intermediary weighted 2-means clustering point is necessary. That is, what if we directly interpolate between our Fano coding tree and balanced 2-means clustering? The results for this are shown in the green curve, which utilizes the marginal frequency based $\widetilde{f}$, and does not interpolate through our intermediary weighted 2-means clustering point. Where the orange curve is a modification to the blue curve, the red curve is a similar modification of the green curve. For this red curve, we interpolate directly from a Fano coding tree from a greedily constructed $\widetilde{f}$ to a balanced 2-means clustering tree. Note that the red curve almost fully overlays the orange one.

We see that while these 4 schemes perform very similarly in terms of their precision at 1, their precision at 3 differs dramatically. The schemes utilizing the greedily constructed $\widetilde{f}$ perform dramatically worse than the schemes using the marginal frequency based $\widetilde{f}$. Additionally, we can see that not utilizing the intermediary weighted 2-means objective yields worse performance (going from the blue to the orange curve).



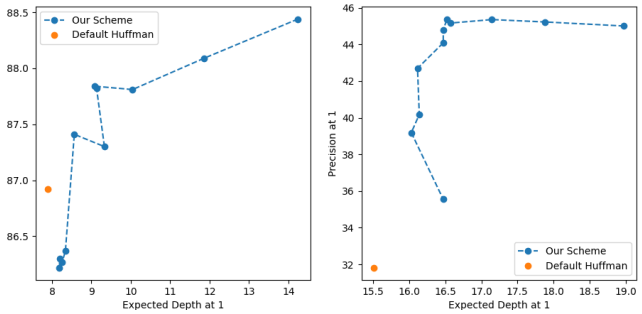
(a) p@1 for Wiki-500K (b) p@3 for Wiki-500K

(c) p@1 for Amazon-670K (d) p@3 for Amazon-670K

Figure 3: Model expected depth versus precision for Parabel augmented with our algorithm.

5.2 Augmented fastText

We augment fastText with our trees for different values of λ and evaluate the performance on the large-scale public XMC datasets AmazonCat-13K and Wiki-500K. We interpolate the full spectrum from fully similarity-based ($\lambda = 0$) to coding-theoretic ($\lambda = 2$). Note that the case of $\lambda = 2$ is equivalent to top-down Fano coding but is not equivalent to fastText's bottom-up Huffman coding. Thus, we plot fastText's default Huffman algorithm for comparison. Figures 4a and 4b show the results on AmazonCat-13K and Wiki-500K, respectively. We observe similar empirical results as in the setting of augmenting Parabel with our algorithm, noting that the expected depths are higher in this setting due to the leaf size of 1 for fastText in contrast to the leaf size of 100 for Parabel.



(a) p@1 for AmazonCat-13K (b) p@1 for Wiki-500K

Figure 4: Model expected depth versus precision@1 for fastText augmented with our algorithm.

5.3 Applying augmented Parabel to e-commerce customer logs

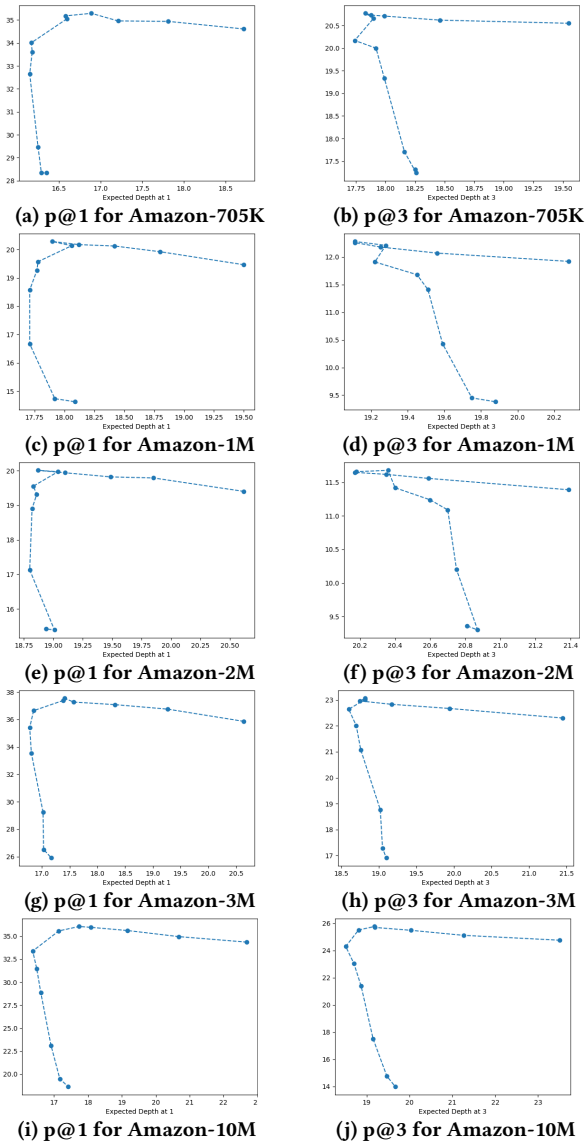
Here we show the effectiveness of our algorithm's augmentation of Parabel in a prototypical e-commerce problem: given a user's context, retrieve a small subset of relevant items from an enormous catalog of products. XMC models are natural candidates for addressing this problem because the inputs (user contexts) are often high-dimensional and the output space (the catalog of products) is discrete, finite, and extremely large.

Dataset	N	d	L	N'	Avg labels/pt
Amazon-705K	951K	207K	705K	50K	2.94
Amazon-1M	1.3M	200K	1.2M	66K	2.94
Amazon-2M	2.4M	486K	2.5M	127K	2.28
Amazon-3M	7.9M	794K	2.7M	364K	3.12
Amazon-10M	29.9M	1.6M	9.9M	1.6M	6.03

Table 2: E-commerce dataset statistics.

To generate each dataset, we aggregate one year of Amazon customer product engagement logs, remove context-product pairs that have low engagement, designate a random 5% of the user contexts as test data, and then use the remaining 95% as training data. We represent the user contexts as sparse high-dimensional embeddings. As seen in Table 2, we generate datasets ranging from 705 thousand to 10 million products (labels) in order to demonstrate the effectiveness of our algorithm in e-commerce datasets of varying orders of magnitude. Figure 5 shows the results.

On all these e-commerce datasets, we observe that our algorithm reduces expected depth at 1 by at least 10% while maintaining precision at 1. On the largest dataset, our algorithm is able to maintain precision at 1 while reducing expected depth by 20%. We similarly observe, to a lesser degree, that our algorithm can reduce expected depth at 3 while marginally improving precision at 3. Given the large scale of modern e-commerce services, as provided by Amazon and others, these efficiency gains could lead to significant reductions in the infrastructure costs of using tree-based XMC models.



(a) p@1 for Amazon-705K (b) p@3 for Amazon-705K

(c) p@1 for Amazon-1M (d) p@3 for Amazon-1M

(e) p@1 for Amazon-2M (f) p@3 for Amazon-2M

(g) p@1 for Amazon-3M (h) p@3 for Amazon-3M

(i) p@1 for Amazon-10M (j) p@3 for Amazon-10M

Figure 5: Expected depth versus precision for Parabel augmented with our algorithm

5.4 Experimental details

The code used in the numerical results is proprietary and cannot be released to the public. However, we provide details in this section with the aim of making our results reproducible.

5.4.1 Datasets. Section 5.1 and Section 5.2 use the train and test datasets provided in [4]. Section 5.1 uses the accompanying bag-of-words input features, whereas Section 5.2 uses the accompanying raw text features.

5.4.2 Models and hyperparameters. Section 5.1 and Section 5.3 use a proprietary implementation of XR-LINEAR [31], which can be viewed as a generalization of Parabel, and we use the suggested settings to recover the special case of Parabel. In particular, we use XR-LINEAR with positive instance feature aggregation (PIFA) label representations, teacher forcing negative sampling, and squared hinge loss. We replace the clustering algorithm with our own, which is equivalent to Parabel's when $\lambda = 0$. For the hyperparameters of Parabel, we set the number of trees to 1 and use the default recommendations for the remaining hyperparameters; i.e., 10 for the maximum number of paths that can be traversed in a tree at prediction time, 100 for the maximum number of labels in a leaf, and 1 for the misclassification penalty for all nodes.

Section 5.2 uses the open source implementation of fastText [1] with the minimal modifications that are needed to use a custom tree in the hierarchical softmax. We use default fastText hyperparameters with the following changes: hierarchical softmax for the loss function, 0.5 for the learning rate, 200 for the number of epochs, 128 for the dimension of the embeddings, 2 for the minimal number of word occurrences, and 2 for the max length of word n-grams.

5.4.3 Our clustering algorithm. For Section 5.1 and Section 5.3, our clustering algorithm was applied recursively until no more than 100 labels remained. The remaining labels then formed a leaf node. For Section 5.2, our clustering algorithm was applied recursively until no more than 1 label remained.

In Section 5.1, we ran our algorithm for $\lambda' = 0, 0.5, 1, 2, 6, 10, 30, 100, 300, 1000, 100000$ where λ is obtained as $\lambda = 2\lambda'/(1+\lambda')$. In Section 5.2, we ran our algorithm for $\lambda' = 0, 0.25, 0.5, 0.75, 1, 2, 4, 16, 64, 256, 100000$. In Section 5.3, we ran our algorithm for $\lambda' = 0, 0.25, 0.5, 0.75, 1, 2, 10, 30, 1000, 100000$.

In the implementation of our clustering scheme, we round the $\boldsymbol{\alpha}$ after each iteration of the clustering algorithm. An additive Laplacian smoothing parameter of $\gamma = 0.1$ was used for all simulations.

6 CONCLUSIONS AND FUTURE WORK

In this work we studied the problem of constructing probabilistic label trees that incorporate both frequency and similarity information. While state of the art schemes ignore one of the two, yielding suboptimal statistical performance or latency, we designed a practical and efficient algorithm for generating a PLT that utilizes both label similarity and frequencies. Our scheme provides a knob to trade off between computational efficiency and statistical performance, which was not previously possible. This approach has promising

[1] https://github.com/facebookresearch/fastText

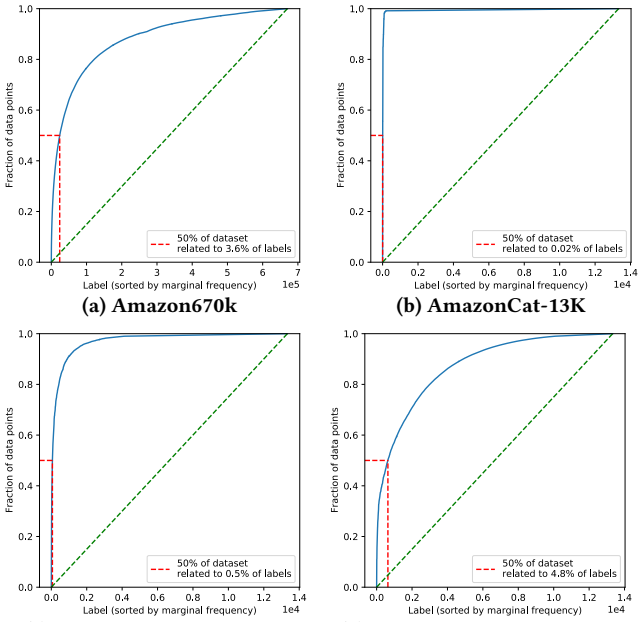
empirical performance on both public datasets and those derived from e-commerce customer logs, and provides a novel theoretical bridge between these two extremes.

One important direction of future work is realizing these expected depth gains as latency improvements in online applications. The focus of this work was on constructing a PLT with low expected depth, meaning that the relevant labels that a standard beam search algorithm returns are high up in this tree. Translating these expected depth gains into wall-clock improvements would require modifying the beam search procedure to use an adaptive stopping condition, returning results before finishing traversing all its paths. Such an adaptive stopping condition would require application-specific parameters, and yield highly implementation dependent latency improvements, and so to compare against implementations of Parabel and fastText we used a standardized metric of expected depth. Developing and analyzing this modified beam search is a critical line of future work towards realizing these expected depth gains as deployable latency improvements.

Appendices

A ADDITIONAL NUMERICAL RESULTS

Below, we include analogues of Figure 1 for Amazon-670K (Figure 6a) and AmazonCat-13K (Figure 6b).



(a) Amazon670k (b) AmazonCat-13K

(c) 3 labels, AmazonCat-13K (d) All labels, AmazonCat-13K

Figure 6: Label frequency imbalance on public datasets.

We can also measure the label imbalance by not just desiring one label per data point, but instead requiring 3 relevant labels per data point (to obtain 100% recall at 3), or require that all relevant labels for a given data point be included in our subset. We see in Figure 6(c,d) how this looks for AmazonCat-13K.

B CONSTRUCTING FREQUENCY VECTOR $\widetilde{f}$

For the case of $k = 1$, we can construct a frequency vector for our tree construction scheme by greedily finding the most frequent label in the dataset, assigning it a frequency proportional to the number of contexts it appears in, and then removing that label and its associated contexts. This is formalized in Algorithm 1. However, for $k > 1$, this seems to be a combinatorially hard problem, for which generating an efficient solution is (to our knowledge) an open problem. In light of this, we construct these frequencies for $k > 1$ by the same greedy scheme mentioned above, and call them $\widetilde{f}$ to denote them as the vector of frequencies that we use for our Fano tree. This is to distinguish them from our vector of marginal label frequencies $\boldsymbol{f}$, which need not be good at minimizing expected depth. In detailing our algorithm, we utilize the label matrix $Y \in \{0,1\}^{N \times L}$, where $Y_{i,\ell} = 1$ if label ℓ is relevant for data point i.

Algorithm 1 `Greedy` $\widetilde{f}$ `construction`

Input: label matrix Y, context frequencies $\boldsymbol{p}$
$\widetilde{\boldsymbol{f}} \leftarrow \boldsymbol{0}$ ▷ initialize all 0s
while $Y^\top \boldsymbol{p} \neq 0$ **do**
  $i^* = \operatorname{argmax}_i Y^\top \boldsymbol{p}$
  $\widetilde{f}_{i^*} = [Y^\top \boldsymbol{p}]_{i^*}$
  $\boldsymbol{p}_{Y(i^*)} = \boldsymbol{0}$ ▷ contexts that relate to label i^*
end while
return $\widetilde{\boldsymbol{f}}$

In practice, another approach for constructing $\widetilde{f}$ has similar performance, and a similarly intuitive justification. We begin by taking our marginal frequency vector $\boldsymbol{f} \propto Y^\top \mathbb{1}$, and sorting these label frequencies by decreasing frequency to get a ranking of labels from most to least frequent. We then iterate over the contexts, and for each context assign its frequency to the most frequent label which it contains. This is detailed more formally in Algorithm 2. This can be softened, as the standard vector $\boldsymbol{f}$ can be constructed by, for each context, adding its frequency to each label it contains. For this extreme $\widetilde{f}$ we assign all of the context's frequency just to the most common label, but we can smoothly interpolate between the two, trading off between uniformly adding the frequency to all relevant labels and just the most common.

Algorithm 2 `Marginal` $\widetilde{f}$ `construction`

Input: label matrix Y, context frequencies $\boldsymbol{p}$
$\widetilde{\boldsymbol{f}} \leftarrow \boldsymbol{0}$ ▷ initialize all 0s
$\boldsymbol{f} = Y^\top p$
for $i = 1, \ldots, n$ **do**
  $j^* = \operatorname{argmax}_j Y_j \odot \boldsymbol{f}$ ▷ $\odot$ is entrywise multiplication
  $\widetilde{f}_{j^*} = \widetilde{f}_{j^*} + p_{j^*}$
end for
return $\widetilde{\boldsymbol{f}}$

Comparing these constructions of $\widetilde{f}$ with $\boldsymbol{f}$, we see that we are simply zeroing out many coordinates, and reducing the value of some others, leading to a more non-uniform distribution.

One further interesting note is that in many real world XMC applications like Dynamic Search Advertising, we not only have label frequencies, but also context frequencies. That is, some contexts are much more common than others. For this work we focused on a uniform distribution over contexts, but all methods we propose can be extended to this scenario with context frequencies, as denoting these with $\boldsymbol{p}$ we can instead construct $\boldsymbol{f} \propto Y^\top \boldsymbol{p}$.

REFERENCES

[1] 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 *Proceedings of the 22nd international conference on World Wide Web*. 13–24.

[2] 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* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 721–729.

[3] Alberto Bertoni, Massimiliano Goldwurm, Jianyi Lin, and Francesco Saccà. 2012. Size constrained distance clustering: separation properties and some complexity results. *Fundamenta Informaticae* 115, 1 (2012), 125–139.

[4] 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

[5] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[6] Róbert Busa-Fekete, Krzysztof Dembczynski, Alexander Golovnev, Kalina Jasinska, Mikhail Kuznetsov, Maxim Sviridenko, and Chao Xu. 2019. On the computational complexity of the probabilistic label tree algorithms. *arXiv preprint arXiv:1906.00294* (2019).

[7] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. 2019. X-Bert: extreme multi-label text classification with using bidirectional encoder representations from transformers. *arXiv preprint arXiv:1905.02331* (2019).

[8] Robert M Fano. 1949. *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics.

[9] Qixuan Huang, Yiqiu Wang, Tharun Medini, and Anshumali Shrivastava. 2018. Extreme Classification in Log Memory. *arXiv preprint arXiv:1810.04254* (2018).

[10] David A Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101.

[11] Kalina Jasinska, Krzysztof Dembczynski, Robert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. 2016. Extreme F-measure Maximization using Sparse Probability Estimates *(Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1435–1444.

[12] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* (2016).

[13] Sujay Khandagale, Han Xiao, and Rohit Babbar. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning* 109, 11 (2020), 2099–2119.

[14] Stanislav Krajči, Chin-Fu Liu, Ladislav Mikeš, and Stefan M Moser. 2015. Performance analysis of Fano coding. In *2015 IEEE International Symposium on Information theory (ISIT)*. IEEE, 1746–1750.

[15] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-Label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 115–124.

[16] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. 165–172.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[19] Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*. 1081–1088.

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

[21] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. 2015. LSHTC: A Benchmark for Large-Scale Text Classification. arXiv:1503.08581 [cs.IR]

[22] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 441–449.

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

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

[25] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *arXiv preprint arXiv:1405.5869* (2014).

[26] Sudheendra Vijayanarasimhan, Jonathon Shlens, Rajat Monga, and Jay Yagnik. 2014. Deep networks with large output spaces. *arXiv preprint arXiv:1412.7479* (2014).

[27] Zhixuan Yang, Chong Ruan, Caihua Li, and Junfeng Hu. 2017. Optimize Hierarchical Softmax with Word Similarity Knowledge. *Polibits* 55 (2017), 11–16.

[28] Ian E.H. Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Halifax, NS, Canada) *(KDD '17)*. Association for Computing Machinery, New York, NY, USA, 545–553.

[29] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. PD-Sparse : A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification *(Proceedings of Machine Learning Research, Vol. 48)*. PMLR, New York, New York, USA, 3069–3077.

[30] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2018. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *arXiv preprint arXiv:1811.01727* (2018).

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