Beyond Standard Performance Measures in Extreme Multi-label Classification

Erik Schultheis
Aalto University
Helsinki, Finland
erik.schultheis@aalto.fi

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

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

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

ABSTRACT

Extreme multi-label classification (XMLC) is the task of selecting, for a given instance, a small subset of relevant labels from a very large set of possible labels. XMLC datasets are characterized by having a long-tailed label distribution, meaning that most of the labels have very few positive instances. With standard performance measures such as precision or nDCG at k, a classifier can ignore a significant portion of the tail labels completely and still get reasonably good performance. However, it is often argued that good predictions in the tail are more "interesting" or "rewarding," yet as of now the XMLC community does not have a way to formalize what this means, nor a set of performance metrics that evaluate this in a principled manner. This paper aims at starting this discussion, first by providing a list of potential performance metrics to be used, as well as some scenarios from which we might infer a more specific meaning of "rewarding." Second, we provide a preliminary investigation into one such metric, *coverage*, and present an efficient greedy strategy aiming to maximize it. A short empirical evaluation shows, that the proposed approach achieves very good results on the measure.

CCS CONCEPTS

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

KEYWORDS

extreme classification, multi-label classification, recommendation, performance measures, coverage

*This work was done during the author's internship in Yahoo!, Paris, France.
†Also with Poznan University of Technology.

KDD '22 OARS Workshop, August 14, 2022, Washington, DC, USA.

ACM Reference Format:
Erik Schultheis, Marek Wydmuch, Rohit Babbar, and Krzysztof Dembczyński. 2022. Beyond Standard Performance Measures in Extreme Multi-label Classification. In *The 2nd International Workshop on Online and Adaptive Recommender Systems, held in conjunction with KDD'22 (KDD '22 OARS Workshop).* ACM, New York, NY, USA, 11 pages.

1 INTRODUCTION

Extreme multi-label classification (XMLC) is the task of selecting, for a given instance, a small subset of relevant labels from a very large set of possible labels. This can be interpreted as an instance-wise bipartite ranking, with the predicted labels being items deemed relevant to a user, ranked above the irrelevant items. As such, XMLC has been applied to various domains such as tagging of text documents [Dekel and Shamir 2010], content annotation for multimedia search [Deng et al. 2011], and a lot of diverse types of recommendation problems, such as webpages-to-ads [Beygelzimer et al. 2009], ads-to-bid-words [Agrawal et al. 2013; Prabhu and Varma 2014], users-to-items [Weston et al. 2013; Zhuo et al. 2020], queries-to-items [Medini et al. 2019], or items-to-queries [Chang et al. 2020].

Because of the nature of its applications, the typical approach in XMLC is to produce a ranking of all labels[1] for a given instance x, and then predict as relevant the k highest-ranked labels. Correspondingly, standard performance metrics in current XMLC are evaluated "at k." Precision at k, denoted P@k, measures which fraction of the top-k ranked labels are actually relevant, without considering the rank at which errors are made. In contrast, (normalized) discounted cumulative gain at k, denoted nDCG@k, gives a larger penalty to prediction errors at the top ranks than to lower ranks. If there are more than k correct labels for the given instance, neither metric is influenced by the choice of the k labels out of the corrected ones.

However, such distinctions might be of great practical interest. The label distribution in XMLC is strongly long-tailed, meaning that most labels are relevant only to very few instances. Consequently, a classifier that never predicts any tail labels can still achieve good scores on the metrics mentioned above Wei and Li [2020]. As such, we would be interested in metrics that prefer "rewarding" [Ye et al.

[1]This does not mean that *every* label needs to be evaluated: Hierarchical approaches might first perform ranking over label clusters, and do ranking on the individual label level only for the most promising label clusters, drastically reducing the computation requirements [Jasinska et al. 2016; Prabhu et al. 2018b].

2020], "diverse" [Babbar and Schölkopf 2019], "rare and informative" [Prabhu et al. 2018a] labels over frequently-occurring head labels. However, designing such a metric is not an easy task. Many measures were proposed as proxies for these vague concepts in a related field of recommendation systems [Castells et al. 2011; Kunaver and Požrl 2017]. None of them gained the title of definitive one. Currently, the XMLC community attempts to capture this using *propensity-scored* variations of the metrics above. These metrics have been derived in order to address the problem of missing labels in XMLC, but because it is commonly assumed that tail-labels are missing more often than head labels, they implicitly give more weight to the tail.

Unfortunately, even though their derivation as unbiased losses under a given propensity model is sound [Jain et al. 2016], their use for emphasizing tail performance seems to be ad-hoc. In particular, if the propensity-model actually captures the missing labels well, then the propensity-scored metrics will just be unbiased estimates of metrics that do *not* give more weight to the tail. A detailed discussion of propensity-scored metrics, and issues in their current usage, can be found in Schultheis et al. [2022].

The aim of this paper is twofold: first, we want to stimulate discussion as to what vague concepts like "rewarding," "diverse," and "informative" mean in XMLC, and how they might be formalized. We present a variety of different existing performance measures that might be candidates for this purpose. Second, we provide some preliminary results, including a greedy prediction strategy, on one of these candidates — the *coverage* metric, which has been previously used as a proxy to illustrate that propensity-scored metrics lead to better predictions of tail labels. Upon closer examination, however, there are several subtleties involved in the definition of coverage. In this work, we present several options, hoping that some of them will catch the attention of the community.

2 SETUP

In this section, we briefly define the notation and setting considered in this paper. We denote with $\mathbb{1}[S]$ the indicator function of the event S, and with $[s] \coloneqq \{1, \dots, s\}$ the set of integers from 1 to s. Capital letters are used to indicate random variables, and bold font for matrices and vectors.

Let $x \in \mathcal{X}$ denote an input instance, associated with a (possibly empty) subset of labels $\mathcal{L}(x) \subset [m]$, called the set of *relevant* or *positive* labels, with the complement, $[m] \setminus \mathcal{L}(x)$, of the *irrelevant* or *negative* ones. The relevant labels are identified by the binary vector $\boldsymbol{y} \in \{0,1\}^m$, such that $y_j = \mathbb{1}[j \in \mathcal{L}(x)]$.

We want to evaluate a scorer h that maps an input instance x to an vector of scores $\hat{\boldsymbol{y}} \in \mathbb{R}^m$. This leads to the top-k ranked predictions, given as a k-tuple of integers $\operatorname{top}_k(\hat{\boldsymbol{y}}) \in \mathbb{N}^k$, and denote with $\mathrm{r}_i(\hat{\boldsymbol{y}}) \in \mathbb{N}$ the rank of the i-th value in $\hat{\boldsymbol{y}}$. In this notation, the classical performance measures[2] such as precision, recall, and nDCG are given by

$$\mathrm{P@}k(\boldsymbol{y}, \hat{\boldsymbol{y}}) \coloneqq k^{-1} \sum_{j \in \operatorname{top}_k(\hat{\boldsymbol{y}})} y_j\,, \quad \mathrm{R@}k(\boldsymbol{y}, \hat{\boldsymbol{y}}) \coloneqq \|\boldsymbol{y}\|_1^{-1} \sum_{j \in \operatorname{top}_k(\hat{\boldsymbol{y}})} y_j\,, \tag{1}$$

$$\mathrm{nDCG@}k(\boldsymbol{y}, \hat{\boldsymbol{y}}) \coloneqq \sum_{j \in \operatorname{top}_k(\hat{\boldsymbol{y}})} \frac{y_j}{\log(\mathrm{r}_j(\hat{\boldsymbol{y}}) + 1)} \Bigg/ \sum_{j=1}^{k} \frac{1}{\log(j+1)}\,. \tag{2}$$

The instance-wise losses above can be defined for a single observation being a pair of x and $\boldsymbol{y}$. Other loss functions can only be defined on the level of a set of observations. We denote a set of n observations $(X^n, \boldsymbol{Y}^n)$, being i.i.d. random variables, with $X^n \coloneqq \{X^{(i)}\}_{i=1}^n$, $\boldsymbol{Y}^n \coloneqq \{\boldsymbol{Y}^{(i)}\}_1^n$. Their realizations are written as $x^n \coloneqq \{x^{(i)}\}_1^n$ and $\boldsymbol{y}$ as $\boldsymbol{y}^n \coloneqq \{\boldsymbol{y}^{(i)}\}_1^n$, and together from the dataset $\mathcal{D} \coloneqq (x^n, \boldsymbol{y}^n)$.

The loss of a classifier with respect to an instance-wise defined loss ℓ is given by

$$\ell(h, x^n, \boldsymbol{y}^n) = \sum_{i=1}^{n} \ell(\boldsymbol{y}^{(i)}, h(x^{(i)})), \tag{3}$$

where ℓ could be, among others, any of the losses defined above.

3 ALTERNATIVE METRICS FOR XMLC

In this section, we list some existing methods that can be used to define new metrics for XMLC tasks. We do not expect that a single metric will turn out to be the "right" metric to use in all cases. In fact, as discussed in the next section, different scenarios might require different metrics.

3.1 Weighted Losses

A simple way of adjusting metrics to take into account utilities of labels, expressing the importance of the label, is to use weights. By specifying the utility for each correct prediction at a given rank, we get a weight matrix $\boldsymbol{W} \in \mathbb{R}^{m \times k}$. Then the loss is defined through:

$$\ell_{k,\boldsymbol{W}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{j \in \operatorname{top}_k(\hat{\boldsymbol{y}})} y_j \cdot W_{j, \mathrm{r}_j(\hat{\boldsymbol{y}})}. \tag{4}$$

This formula generalizes P@k through

$$W^{\mathrm{P}}_{j,s} = k^{-1}\,, \tag{5}$$

and nDCG@k through,

$$W^{\mathrm{nDCG}}_{j,s} = \frac{1}{c \log(\mathrm{r}_j(\hat{\boldsymbol{y}}) + 1)}, \text{ where } c \coloneqq \sum_{j=1}^{k} \frac{1}{\log(j+1)}\,. \tag{6}$$

The weights can also be defined using a label-wise propensity model introduced by Jain et al. [2016] to deal with missing labels to achieve an unbiased loss. In such a case, the popular propensity-scored P@k, denoted by PSP@k, can also be expressed by (4) with the following weights:

$$W^{\mathrm{PSP}}_{j,s} = \frac{1}{k p_j}\,, \tag{7}$$

where $p_j \coloneqq \mathbb{P}\left[\tilde{Y}_j = 1 \mid Y_j = 1\right]$ is the probability of observing the label j as relevant ($\tilde{Y}_j = 1$) in case it is truly relevant ($Y_j = 1$). Combining the per-label weighting of propensity-scored losses

[2] Often, if these are to be minimized, they are called losses, and if they are maximized utilities. For the purposes of this paper, this distinction does not matter and we are using the term loss for any quantity to be optimized.

with the rank-based weights of nDCG@k results in PSnDCG@k with weights as follows:

$$W^{\text{PSnDCG}}_{j,s} = \frac{1}{c \log(\mathrm{r}_j(\hat{\boldsymbol{y}}) + 1) p_j}, \tag{8}$$

where c is taken from (6). In the missing label interpretation, the PSP@k values are actually just unbiased estimates for regular precision in the clean data case, without any adjustment for tail labels. If, on the other hand, the PSP@k metric is used with the goal of up-weighting the tail labels, which is a common approach in XMLC, then the constants p_j are not chosen in a principled way [Schultheis et al. 2022].

The advantage of such weighted losses is that they would be relatively straight-forward to integrate into existing XMLC pipelines. Their disadvantage is the large number of free parameters, and correspondingly that the resulting values would be difficult to interpret without a good interpretation of $\boldsymbol{W}$.

3.2 Macro-Averaging at k

Better interpretability could be achieved using macro-averaged losses. In this case, a binary classification loss Ψ is calculated for each label separately, and the results are then averaged. Consequently, such a loss can no longer be defined on the level of a single instance, but only on a sample $(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)})_{i=1}^n$ of n realizations. In order to evaluate such a loss, one would first calculate a per-label confusion matrix $\mathbf{C}^j \in \mathbb{R}^{2\times 2}$ as

$$C^j_{1,t} = n^{-1} \sum_{i=1}^{n} \mathbb{1}\left[j \in \operatorname{top}_k(\hat{\boldsymbol{y}}^{(i)}), y^{(i)}_j = t \right], \tag{9}$$

$$C^j_{0,t} = n^{-1} \sum_{i=1}^{n} \mathbb{1}\left[j \notin \operatorname{top}_k(\hat{\boldsymbol{y}}^{(i)}), y^{(i)}_j = t \right]. \tag{10}$$

The macro-averaged loss is then given by

$$\ell_{k,\text{MA}} = m^{-1} \sum_{j=1}^{m} \Psi(\mathbf{C}^j). \tag{11}$$

Examples include *Macro-Precision* with $\Psi(\mathbf{C}) = C_{1,1}/(C_{1,1} + C_{1,0})$, *Macro-Recall* with $\Psi(\mathbf{C}) = C_{1,1}/(C_{1,1} + C_{0,1})$, and *Macro-F-Measure* $\Psi(\mathbf{C}) = (1+\beta^2)C_{1,1}/((1+\beta^2)C_{1,1} + \beta^2 C_{0,1} + C_{1,0})$.

3.3 Abandonment at k and Coverage at k

Another interesting metric, used e.g. in search engines [Radlinski et al. 2008], is Abandonment@k defined as:

$$\text{Abandonment@}k(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \mathbb{1}\left[\exists j \in \operatorname{top}_k(\hat{\boldsymbol{y}}) : y_j = 1 \right]. \tag{12}$$

This measures whether an instance is "abandoned", in the sense that not a single prediction for that instance is relevant. While this untypical formulation enforces diversity in the predicted set, in itself, this metric does not promote tail labels. However, we can consider the corresponding macro-average to this instance-wise metric. This corresponds to $\Psi(\mathbf{C}) = \mathbb{1}\left[C_{1,1} \neq 0\right]$, i.e., there is at least one instance for which the label has been predicted correctly as relevant. The result is a less well known, but still established metric, *coverage*. Expanding the expressions, we can write

$$\text{Cov@}k(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n) = m^{-1} \sum_{j=1}^{m} \mathbb{1}\left[\exists i \in [n] : j \in \operatorname{top}_k(\hat{\boldsymbol{y}}^{(i)}) : y^{(i)}_j = 1 \right]. \tag{13}$$

Coverage has been used in Jain et al. [2016] as one metric to show that optimizing for propensity-scored precision leads to more diverse predictions, in the sense that for more labels there is at least one instance where the label is actually among the correct predictions. It was also later suggested in the literature as a good measure for tail-labels [Babbar and Schölkopf 2019; Wei and Li 2020].

3.4 Multi-Objective Optimization

Even in cases where a clear concept of the desired prediction property, "diversity", exists, one still needs to find an optimization scheme that promotes this property while not sacrificing too much in terms of standard performance.

Constrained Model Selection. One way to achieve this would be to set an acceptable level of standard performance, and then turn the unconstrained optimization problem into a constrained optimization task. The goal is to maximize the diversity measure, while keeping the precision at a required minimal level $\alpha \in (0, 1)$. This task is given by

$$\begin{aligned} \operatorname{argmax}_{h\in\mathcal{H}} \quad & \mathbb{E}\big[\text{diversity}[h, X^n, Y^n]\big] \\ \text{s.t.} \quad & \text{P@}k[h] \geq \alpha\,. \end{aligned} \tag{14}$$

As a concrete example, one could consider the entropy of the predictions: Define the distribution of correctly predicted labels of a classifier h through

$$\rho_j[h] \coloneqq C^{(j)}_{1,1} \Big/ \sum_{j'=1}^{m} C^{(j)}_{1,1}. \tag{15}$$

In such a case, the (empirical version of the) maximization problem would be

$$\begin{aligned} \operatorname{argmax}_{h\in\mathcal{H}} \quad & -\sum_{j=1}^{m} \rho_j[h] \log\big(\rho_j[h]\big) \\ \text{s.t.} \quad & \text{P@}k[h] \geq \alpha\,. \end{aligned} \tag{16}$$

Aggregation. An alternative is to combine the different objectives to a combined score using some interpolation technique, for example a (weighted) arithmetic or harmonic mean.

Such an approach was taken by Bradley and Smyth [2001], who define the diversity of a set of labels in terms of a similarity metric similarity : $\mathcal{Y} \times \mathcal{Y} \longrightarrow [0, 1]$ between two labels via[3]

$$\text{diversity}(\mathsf{y}_1, \ldots, \mathsf{y}_k) = \sum_{i=1}^{k} \sum_{j=i}^{k} \frac{1 - \text{similarity}(\mathsf{y}_i, \mathsf{y}_j)}{k(k-1)/2}. \tag{17}$$

Of course, such a definition just shifts the difficulty of defining "diversity" to defining "similarity" of labels.

[3] We have translated the description in the original paper into XMLC notation — in particular, "items" c in recommender systems correspond to "labels" y and queries t to instances x in XMLC.

Table 1: Performance measures (in percent) on AmazonCat-13k with full and partial labels during training.

Metric	full labels			head labels		
	@1	@3	@5	@1	@3	@5
Precision	93.0	78.9	64.1	93.0	76.6	58.1
nDCG	93.0	82.2	71.2	58.1	80.6	66.8
Abandonment	93.0	96.4	97.6	93.0	96.6	97.8
Coverage	24.8	51.5	69.4	6.1	7.4	7.5
Macro-Precision	21.8	39.2	43.3	4.9	4.9	3.8
Macro-Recall	24.3	50.8	69.4	3.1	4.5	7.5
Macro-F1	19.3	39.8	51.3	2.8	3.7	4.9

3.5 Example

As an illustration, we trained a DiSMEC [Babbar and Schölkopf 2017] model on the full AmazonCat-13K [McAuley et al. 2015] dataset as well as on a reduced version where only the 1000 most popular labels are kept in the training set. A similar experiment was performed before by Wei and Li [2020].

As Table 1 shows, the standard performance measures are almost unperturbed by reducing the label space to the head labels, especially at low values of k. In contrast, coverage and macro measures decrease to almost zero if tail labels are ignored.

4 SCENARIOS

This section gives an (incomplete) list of scenarios in which XMLC methods can be applied and discusses which notions of "rewarding" labels might apply in these circumstances, as well as which type of loss function this could correspond to. If you have a use-case not covered here, we would be delighted to hear from you.

4.1 Document Tagging and Indexing

One typical application of XMLC is for document tagging, e.g., the standard benchmark datasets Wikipedia-500K, and Eurlex-4K are tasks in which a document is mapped to a (small) set of tags or categories to which the document belongs. We think that there are two main ways in which one relevant label can be considered more "rewarding" than another:

Avoiding Redundancy. The tags available for Wikipedia articles are parts of (potentially multiple) hierarchies, with specific labels such as "Italian Watercolourists" being a refinement of "Italian Painters", which in turn is a subcategory of "Italian Artists". As a user of Wikipedia, adding the relevant label "Italian Artists" to an article already annotated with "Italian Painters" does not convey any additional information; the super-category label is redundant. Thus, one goal for rewarding labels could be to select the k relevant labels that have the least redundancy.

Indexing. One advantage of having tagged documents is that the tags allow building of an index. If indexing is the primary goal, then the performance metric used to evaluate the tagging should reflect that. This naturally leads to *macro-averaged* metrics: Each index page corresponds to a label, for which the classification metric is evaluated, so the average quality of an index page is given by the macro average.

Offline Prediction. We expect document tagging to typically have the option of being run offline, which allows predictions to be made at the population level.

4.2 Related Queries and Ads

A second typical use case is to find related queries or relevant ads for a given user input. Here we have three perspectives to consider:

Covering Users. First, one might be interested in ensuring that there is *at least one* relevant suggestions for each user, in order to avoid an unsatisfied user leaving the service. This corresponds to the *abandonment* metric discussed above.

Covering Labels. On the other hand, it might also be important to ensure that e.g., each possible advertisement has at least some user input for which it would be displayed. This corresponds to the *coverage* metric.

Cost-sensitive Predictions. It also might be the case that there is naturally a different, quantifiable benefit of presenting one relevant label over another, for example, if one advertisement provider were to pay more per click than another. A streaming service might, at the same level of relevance, prefer to present its own content over licensed third-party content. In this case, the weighted losses discussed above would be applicable.

Online Prediction. We generally expect these scenarios to require predictions of an online nature, which means that decisions have to be made at the instance level, possibly with the aid of some collected global statistics.

5 RESULTS ON COVERAGE AT K

The coverage at k become one of the top suggestions for measuring performance on tail-labels in XMLC [Babbar and Schölkopf 2019; Jain et al. 2016; Wei and Li 2020]. However, to the best of our knowledge, there were no attempts to optimize it directly in XMLC. In this section, we shortly formalize the problem of selecting k labels for each instance of the given dataset in order to optimize Cov@k and show that this is an NP problem. Because of that, we propose a heuristic — a greedy method that can be applied as a plug-in approach to any probabilistic classifier. Additionally, we show how to combine it with probabilistic label trees, a popular approach for XMLC problems for efficient inference. We demonstrate the attractiveness of this approach in a wide empirical study on popular XMLC benchmark datasets.

5.1 Optimizing Coverage at k

As a first step, we want to determine which k labels to select for prediction if we knew the distribution of labels, i.e. we want to find the Bayes classifier among all hypotheses that predict exactly k labels for each instance.

$$\begin{aligned} &\max_{\hat{\mathbf{z}}^n \in \{0,1\}^{n\times m}} \mathbb{E}\left[\mathrm{Cov@}k(\boldsymbol{Y}^n, \hat{\mathbf{z}}^n) \mid X^n = x^n\right] \\ &\text{s.t. } \forall i \in [n] : \|\hat{\mathbf{z}}^{(i)}\|_1 = k\,. \end{aligned} \tag{18}$$

Algorithm 1 Greedy Algorithm for Cov@k(x^n, k)

```
f_j ← 1 for all j ∈ [m]                          ▷ Initialize a vector f for calculation of a probability of a "failure"
for x^(i) ∈ x^n do                               ▷ For all the examples in the dataset
    for j ∈ [m] do                               ▷ For all labels
        g_j ← f_j − (1 − η̂_j(x^(i))) f_j = f_j η̂_j(x^(i))   ▷ For expected gain g_j from selecting the label j
    ẑ^(i) ← select top-k(w)                      ▷ Select k labels with the highest g_j
    for j ∈ [m] : ẑ_j^(i) = 1 do                 ▷ For k selected labels
        f_j ← (1 − η̂_j(x^(i))) f_j               ▷ Update expected probability of "failure" f_j
```

To obtain this, we first rewrite equation (13) for Cov@k in a way that makes it easier to apply probabilistic reasoning:

$$\begin{aligned}
\text{Cov@}k(\boldsymbol{y}^n, \hat{\boldsymbol{z}}^n) &= m^{-1} \sum_{j=1}^{m} \mathbb{1}\left[\exists i \in [n] : j \in \text{top}_k(\hat{\boldsymbol{z}}^{(i)}) : y_j^{(i)} = 1\right] \\
&= m^{-1} \sum_{j=1}^{m} 1 - \mathbb{1}\left[\forall i \in [n] : j \in \text{top}_k(\hat{\boldsymbol{z}}^{(i)}) : y_j^{(i)} = 0\right] \\
&= m^{-1} \sum_{j=1}^{m} \left(1 - \prod_{i=1}^{n} \left(1 - y_j^{(i)} \mathbb{1}\left[j \in \text{top}_k(\hat{\boldsymbol{z}}^{(i)})\right]\right)\right) \\
&= 1 - m^{-1} \sum_{j=1}^{m} \left(\prod_{i=1}^{n} \left(1 - y_j^{(i)} \mathbb{1}\left[j \in \text{top}_k(\hat{\boldsymbol{z}}^{(i)})\right]\right)\right). \qquad (19)
\end{aligned}$$

Because $\forall i \in [n] : \|\hat{\boldsymbol{z}}^{(i)}\|_1 = k$, we can replace the $\mathbb{1}\left[j \in \text{top}_k(\hat{\boldsymbol{z}}^{(i)})\right]$ operation with multiplication:

$$\text{Cov@}k(\boldsymbol{y}^n, \hat{\boldsymbol{z}}^n) = 1 - m^{-1} \sum_{j=1}^{m} \left(\prod_{i=1}^{n} \left(1 - y_j^{(i)} \hat{z}_j^{(i)}\right)\right). \qquad (20)$$

Plugging this into the expectation to obtain the Bayes-classifier, we can use the linearity of expectations to get

$$\begin{aligned}
&\mathbb{E}\left[\text{Cov@}k\left(Y^n, \hat{\boldsymbol{z}}^n\right) \mid X^n\right] \\
&\quad = 1 - m^{-1} \sum_{j=1}^{m} \mathbb{E}\left[\prod_{i=1}^{n} \left(1 - Y_j^{(i)} \hat{z}_j^{(i)} \mid X^n\right)\right]. \qquad (21)
\end{aligned}$$

Because for any fixed j, the labels $\{Y_j^{(i)}\}_1^n$ for different instances are independent according to our i.i.d. assumption on the sample,

$$\begin{aligned}
&\mathbb{E}\left[\text{Cov@}k\left(Y^n, \hat{\boldsymbol{z}}^n\right) \mid X^n\right] \\
&\quad = 1 - m^{-1} \sum_{j=1}^{m} \prod_{i=1}^{n} \left(1 - \mathbb{E}\left[Y_j^{(i)} \hat{z}_j^{(i)} \mid X^n\right]\right) \\
&\quad = 1 - m^{-1} \sum_{j=1}^{m} \prod_{i=1}^{n} \left(1 - \mathbb{P}\left[Y_j^{(i)} \mid X^n\right] \hat{z}_j^{(i)}\right). \qquad (22)
\end{aligned}$$

Notice that the $\prod_{i=1}^{n} \left(1 - \mathbb{P}\left[Y_j^{(i)} \mid X^n\right] \hat{z}_j^{(i)}\right)$ correspond to the probability of j being irrelevant for all instances it was selected for.

Theorem 5.1 (Bayes Classifier for Coverage at k). *The optimal prediction $\hat{\boldsymbol{z}}_*^n$, i.e. the Bayes classifier, is now characterized by*

$$\begin{aligned}
&\min_{\hat{\boldsymbol{z}}^n \in \{0,1\}^{n \times m}} m^{-1} \sum_{j=1}^{m} \prod_{i=1}^{n} \left(1 - \eta_j^{(i)} \hat{z}_j^{(i)}\right) \\
&\quad \text{s.t. } \forall i \in [n] : \|\hat{\boldsymbol{z}}^{(i)}\|_1 = k, \qquad (23)
\end{aligned}$$

where $\eta_j^{(i)} := \mathbb{P}\left[Y_j^{(i)} \mid X^n\right] = \mathbb{P}\left[Y_j^{(i)} \mid X^{(i)}\right]$ is the conditional probability of the label for each instance X.

This shows that to optimize Cov@k it is enough to know conditional probabilities of labels, which can be estimated using many types of multi-label classifiers, such as decision trees, k-nearest neighbors, or binary relevance trained with proper composite surrogate losses, e.g., squared error, squared hinge, logistic or exponential loss [Agarwal 2014; Zhang 2004].

Unfortunately solving the optimization problem stated above to find out which labels to score as top-k is non-trivial. The naive brute force approach requires to consider $\binom{m}{k}^n$ possibilities. Let us notice that if $\forall i \in [n] : \eta^{(i)} \in \{0,1\}^m$ the problem of maximizing the Cov@k reduces to a variant of maximum coverage problem [Coffman et al. 1996], which is unfortunately an NP-hard problem and thus we cannot expect to find the optimal solution for the Cov@k efficiently.

5.2 Greedy approach

However, we propose a simple greedy algorithm for optimizing Cov@k that works in linear time with respect to m. The proposed approach is presented in Algorithm 1. The general idea of the greedy algorithm is that for the currently considered instance $x^{(i)}$ we want to select k labels that maximize the gain on the Cov@k in the current step, taking into account past decisions. In this case, for each instance, we select the k labels which the most decrease the probability that the label is irrelevant for all instances for which it was selected, according to the estimates of conditional probability $\eta_j^{(i)}$ denoted by $\hat{\eta}_j(x^{(i)})$. This corresponds to the greedy optimization of the product part of the Equation (22). It is equivalent to selecting top-k labels, which the most increase the probability that the label is relevant for at least one instance it was selected for. Next, the estimates of the probabilities that the given label j is irrelevant for all instances it was previously selected for, denoted as f_j, are updated for selected labels, and the procedure is repeated for the next instance.

5.3 Weighted variant of greedy approach

One can notice that the proposed greedy procedure will select likely labels at the beginning. However, once the probability of these being covered reach a high value (low f_j values), it will start selecting less and less likely (mostly tail) labels as its prediction, which might cause an undesired drop in performance on measures like precision at k. Ideally, one would like to control the trade-off between optimizing for coverage and precision at k. This can be easily achieved by modifying Algorithm 1, for example, by adding

constant β to the all current values of f_j, resulting in the gain function $g_j := (f_j + \beta)\hat{\eta}_j(x^{(i)})$, with β controlling the trade-off. Choosing $b = 0$ gives the original greedy algorithm, and large values of β prioritize standard top-k prediction according to conditional probabilities.

5.4 Efficient greedy approach

To use the proposed greedy approach, one needs first to obtain marginal conditional probabilities for all the labels for a given instance (linear complexity with respect to m). This might be problematic in the setting of extreme classification. Fortunately, the top-k labels with the highest "gain" on Cov@k can be found efficiently when using Probabilistic Label Trees (PLT) [Jasinska et al. 2016], a popular approach to XMLC, being the core of many existing state-of-the-art algorithms (e.g., Parabel [Prabhu et al. 2018b], extremeText [Wydmuch et al. 2018], Bonsai [Khandagale et al. 2019], AttentionXML [You et al. 2019], X-Transformers [Chang et al. 2020] methods). PLTs factorize the conditional probability $\eta_j^{(i)}$ of a label j using the chain rule and use probabilistic classifiers on the path from the tree root to the leaf corresponding to that label to estimate the factors.

The tree structure allows for an inference procedure that uses a tree-search approach like uniform-cost search, or beam-search [Russell and Norvig 2009] to efficiently find top-k estimates $\hat{\eta}_j^{(i)}$. Recently Wydmuch et al. [2021] proposed weighted variant of the inference procedure, that uses A^*-search [Russell and Norvig 2009] based algorithm for efficiently finding the k leaves with highest value of $w_j\hat{\eta}_j(\boldsymbol{x})$, where $w_j \in [0, \infty)$ is the given weight for label j. This method was originally used for optimal prediction for PSP@k, with inverse-propensity p_j^{-1} as weights for all labels $j \in [m]$. We use the same procedure to efficiently search a tree for the top-k labels with highest value of $g_j = f_j\hat{\eta}_j(x^{(i)})$, where f_j serve as weights for all $j \in [m]$. This approach is also compatible with the proposed weighted variant with β parameter. We present the details of the method in the appendix.

5.5 Experimental comparison

Here we demonstrate that even a simple greedy method achieves much higher scores on Cov@k than other inference procedures. We compare a few variants of PLTs, one with inference optimal for standard precision@k [Wydmuch et al. 2018] (PLT), one optimal for propensity-scored precision@k [Wydmuch et al. 2021] (PS-PLT), and PLT with our proposed greedy approach (COV-PLT). We implemented COV-PLT modifying the recently introduced napkinXC [Jasinska-Kobus et al. 2020] implementation of PLTs[4] which implements both standard PLTs and weighted PLTs. We add to the comparison a simple baseline — Bounded Random Selection method (BRS-PLT), proposed in Bradley and Smyth [2001]. It aims to increase the diversity of prediction by selecting k labels randomly from a larger set of the top-bk labels. In our experiment, we additionally sort sampled labels descending by their estimated conditional probability. We use $b = 2$ (and $k = 5$), since we observed that larger b results in drops in performance on all presented measures. Finally, we also add to the comparison the weighted variant of COV-PLT with parameter β (COV$_\beta$-PLT). Here we present results for $\beta = 0.25$ for all datasets, since it seems to provide good tradeoff between P@k and Cov@k. We include a comparison for different values of β in the appendix.

We conduct a comparison on six well-established XMLC benchmark datasets from the XMLC repository [Bhatia et al. 2016], for which we use the original train and test splits. For predicting and evaluating on PSP@k measure we use values recommended in Jain et al. [2016]. Values of A and B are included in Table 2. We evaluate all algorithms with P@k, PSP@k and Cov@k.

We train all PLTs using the LIBLINEAR library [Fan et al. 2008] with L_2-regularized logistic regression. We use an ensemble of 3 trees built with the hierarchical 2-means clustering algorithm (with clusters of size 100), popularized by Parabel [Prabhu et al. 2018b]. Because the tree-building procedure involves randomness, we repeat all PLTs experiments five times and report the mean performance with standard errors. The experiments were performed on an Intel Xeon E5-2697 v3 2.6GHz machine with 128GB of memory.

The main results of the experimental comparison are presented in Table 2. The COV-PLT obtains much higher results on Cov@k than P@k and PSP@k oriented variants. The BRS-PLT is, in most cases, dominated on all measures by other methods except standard PLT. Additionally, the inference time is comparable to the inference time of PLT, and PS-PLT. However, optimizing for coverage comes at a high cost in precision. As shown, usage of COV$_\beta$-PLT can mitigate that problem. For the selected value of β, most of the performance at P@k is recovered, while the method is still the second best in terms of Cov@k in most cases.

6 DISCUSSION AND OUTLOOK

We have presented a range of potential metrics for long-tailed prediction, and taken a closer look at the specific instance of coverage. We demonstrated that a simple plug-in greedy approach significantly boosts the scores on coverage at k and that it can be applied to the popular probabilistic label trees for efficient inference. There are still some shortcomings that need to be addressed in subsequent work, though, in particular achieving a trade-off between standard performance and coverage.

6.1 Alternative Definitions of Coverage at k

The naive definition of coverage as given in equation (13) has two shortcomings. First, one could define coverage for a standardized sample size, n', to define

$$n'\text{-Cov@}k_{\mathbb{P}} = \mathbb{E}_{X',Y'\sim\mathbb{P}^{n'}}\left[\text{Cov@}k(h(X'), Y')\right] \tag{24}$$

Then, to define an empirical counterpart, given a dataset of size $n > n'$, we evaluate n'-Cov@k as the expectation of classical coverage for picking n' data points from the dataset uniformly at random:

$$n'\text{-Cov@}k_{\mathcal{D}} = \mathbb{E}_{X',Y'\sim\mathcal{U}^n(\mathcal{D})}\left[\text{Cov@}k(X', Y')\right] \tag{25}$$

An intriguing property of this definition is that it allows interpolation between P@k and (classical) Cov@k. For $n' = n$, it corresponds to a bootstrap estimate[5] of Cov@k, and for $n' = 1$ it corresponds

[4] https://github.com/mwydmuch/napkinXC

[5] i.e., resampling the original dataset and taking the average

Table 2: Mean performance with standard errors, rounded to two decimal places, of different variants of PLT, PS-PLT, COV-PLT, BRS-PLT and COV$_\beta$-PLT on standard precision, propensity-scored precision, and coverage@$\{1, 3, 5\}$ [%] and test CPU time per instance [ms]. A and B indicate values of the parameters of the propensity model for estimating values of p_j. The best result for each measure is in bold.

	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	Cov@1	Cov@3	Cov@5	$t_{\text{test}}/n_{\text{test}}$
	EurLex-4K, $A = 0.55$, $B = 1.5$									
PLT	**81.70 ± 0.09**	**68.22 ± 0.12**	57.11 ± 0.11	35.97 ± 0.09	43.22 ± 0.10	47.28 ± 0.13	22.95 ± 0.16	41.85 ± 0.15	52.41 ± 0.13	**3.17 ± 0.06**
PS-PLT	79.19 ± 0.09	67.81 ± 0.07	**57.15 ± 0.09**	**44.73 ± 0.06**	**48.52 ± 0.11**	**50.84 ± 0.11**	36.00 ± 0.14	53.35 ± 0.17	61.33 ± 0.12	6.03 ± 0.15
COV-PLT	62.83 ± 0.17	55.19 ± 0.11	48.12 ± 0.09	41.74 ± 0.11	45.29 ± 0.10	47.31 ± 0.13	**44.77 ± 0.12**	**61.05 ± 0.08**	**66.39 ± 0.17**	6.51 ± 0.09
BRS-PLT	70.44 ± 0.19	50.82 ± 0.15	37.15 ± 0.06	31.93 ± 0.13	33.53 ± 0.14	31.94 ± 0.09	26.35 ± 0.12	43.70 ± 0.30	50.64 ± 0.23	4.63 ± 0.07
COV$_\beta$-PLT	75.64 ± 0.08	66.12 ± 0.13	55.84 ± 0.12	42.06 ± 0.08	47.19 ± 0.11	49.76 ± 0.16	39.84 ± 0.11	56.15 ± 0.15	62.95 ± 0.27	4.67 ± 0.10
	AmazonCat-13K, $A = 0.55$, $B = 1.5$									
PLT	**93.33 ± 0.00**	**78.83 ± 0.02**	**64.11 ± 0.03**	50.02 ± 0.01	63.11 ± 0.01	71.15 ± 0.04	15.88 ± 0.07	42.21 ± 0.07	62.97 ± 0.06	**1.71 ± 0.07**
PS-PLT	88.04 ± 0.05	77.16 ± 0.04	63.84 ± 0.03	**66.81 ± 0.03**	**72.05 ± 0.04**	**74.88 ± 0.05**	69.83 ± 0.05	77.12 ± 0.05	80.26 ± 0.10	4.98 ± 0.49
COV-PLT	56.57 ± 0.14	50.56 ± 0.09	42.85 ± 0.07	57.46 ± 0.12	58.66 ± 0.11	59.63 ± 0.10	**80.07 ± 0.07**	**86.51 ± 0.09**	**88.62 ± 0.10**	13.66 ± 0.98
BRS-PLT	81.44 ± 0.04	56.12 ± 0.03	39.42 ± 0.04	46.54 ± 0.03	48.72 ± 0.04	45.18 ± 0.06	32.78 ± 0.12	60.44 ± 0.14	69.16 ± 0.09	2.95 ± 0.13
COV$_\beta$-PLT	92.28 ± 0.03	78.52 ± 0.05	63.93 ± 0.08	56.41 ± 0.03	66.65 ± 0.05	72.93 ± 0.12	64.67 ± 0.10	74.97 ± 0.08	79.30 ± 0.08	3.80 ± 0.19
	Wiki10-31K, $A = 0.55$, $B = 1.5$									
PLT	**85.52 ± 0.09**	**74.51 ± 0.05**	**64.49 ± 0.04**	12.75 ± 0.01	14.45 ± 0.01	15.11 ± 0.00	2.26 ± 0.01	5.48 ± 0.01	7.80 ± 0.02	**29.40 ± 0.56**
PS-PLT	74.12 ± 0.09	65.87 ± 0.13	59.08 ± 0.15	**21.83 ± 0.07**	**19.77 ± 0.03**	**19.12 ± 0.04**	8.01 ± 0.04	12.80 ± 0.03	15.29 ± 0.03	89.40 ± 1.30
COV-PLT	55.96 ± 0.16	47.05 ± 0.15	41.45 ± 0.11	20.13 ± 0.07	18.31 ± 0.04	17.90 ± 0.06	**9.79 ± 0.05**	**15.95 ± 0.02**	**19.29 ± 0.05**	153.34 ± 0.67
BRS-PLT	76.87 ± 0.18	59.79 ± 0.12	48.39 ± 0.15	12.40 ± 0.05	12.48 ± 0.01	12.26 ± 0.04	2.94 ± 0.01	**6.23 ± 0.03**	**8.59 ± 0.03**	48.17 ± 0.70
COV$_\beta$-PLT	72.31 ± 0.11	65.12 ± 0.06	57.53 ± 0.12	18.35 ± 0.04	17.42 ± 0.04	17.47 ± 0.06	7.90 ± 0.04	12.66 ± 0.04	15.37 ± 0.06	72.82 ± 1.76
	WikiLSHTC-325K, $A = 0.5$, $B = 0.4$									
PLT	63.67 ± 0.20	42.11 ± 0.14	31.23 ± 0.11	25.91 ± 0.09	31.81 ± 0.12	35.48 ± 0.14	18.82 ± 0.01	33.83 ± 0.05	40.40 ± 0.06	**6.13 ± 0.21**
PS-PLT	64.47 ± 0.43	43.19 ± 0.29	**32.08 ± 0.21**	32.84 ± 0.18	36.27 ± 0.24	**39.38 ± 0.27**	27.98 ± 0.05	41.42 ± 0.12	47.38 ± 0.15	17.15 ± 1.02
COV-PLT	55.59 ± 0.30	36.18 ± 0.19	26.71 ± 0.15	**34.81 ± 0.19**	**36.50 ± 0.21**	38.73 ± 0.25	**35.33 ± 0.10**	**47.03 ± 0.14**	**51.69 ± 0.17**	14.32 ± 1.25
BRS-PLT	47.37 ± 0.14	27.42 ± 0.09	19.05 ± 0.07	20.48 ± 0.07	21.60 ± 0.08	22.51 ± 0.10	21.11 ± 0.04	32.76 ± 0.08	37.42 ± 0.11	9.47 ± 0.46
COV$_\beta$-PLT	**65.04 ± 0.25**	**43.33 ± 0.18**	31.91 ± 0.16	31.65 ± 0.13	35.87 ± 0.18	38.97 ± 0.22	29.69 ± 0.05	42.43 ± 0.11	47.94 ± 0.13	8.09 ± 0.02
	WikipediaLarge-500K, $A = 0.5$, $B = 0.4$									
PLT	67.33 ± 0.34	48.15 ± 0.25	37.66 ± 0.19	26.22 ± 0.11	30.87 ± 0.15	34.06 ± 0.18	16.30 ± 0.01	30.58 ± 0.07	37.62 ± 0.10	**33.27 ± 0.74**
PS-PLT	**67.53 ± 0.21**	**48.68 ± 0.15**	**38.23 ± 0.12**	34.12 ± 0.10	**35.70 ± 0.12**	**38.14 ± 0.13**	24.77 ± 0.04	38.41 ± 0.08	44.81 ± 0.09	82.21 ± 7.64
COV-PLT	55.05 ± 0.26	39.29 ± 0.19	30.77 ± 0.15	**34.41 ± 0.17**	35.11 ± 0.18	36.91 ± 0.22	**31.52 ± 0.08**	**44.20 ± 0.13**	**49.55 ± 0.17**	63.29 ± 3.42
BRS-PLT	52.69 ± 0.19	33.61 ± 0.12	24.54 ± 0.10	21.42 ± 0.07	22.08 ± 0.09	22.76 ± 0.10	18.26 ± 0.04	29.96 ± 0.08	35.08 ± 0.10	41.83 ± 3.01
COV$_\beta$-PLT	66.85 ± 0.28	48.32 ± 0.21	37.69 ± 0.18	31.70 ± 0.12	34.40 ± 0.15	36.98 ± 0.19	26.48 ± 0.05	39.43 ± 0.10	45.41 ± 0.14	53.01 ± 0.44
	Amazon-670K, $A = 0.6$, $B = 2.6$									
PLT	**44.88 ± 0.17**	**40.09 ± 0.15**	**36.60 ± 0.15**	26.24 ± 0.09	30.13 ± 0.11	33.72 ± 0.14	11.08 ± 0.02	26.34 ± 0.06	37.94 ± 0.12	**17.90 ± 0.11**
PS-PLT	43.71 ± 0.10	39.72 ± 0.09	36.60 ± 0.10	**31.14 ± 0.07**	**33.45 ± 0.09**	**35.60 ± 0.11**	13.27 ± 0.03	30.42 ± 0.07	41.18 ± 0.11	29.59 ± 0.65
COV-PLT	42.80 ± 0.13	38.75 ± 0.13	35.65 ± 0.14	29.50 ± 0.09	32.39 ± 0.12	34.82 ± 0.15	**14.56 ± 0.04**	**32.08 ± 0.10**	**42.02 ± 0.16**	22.92 ± 0.68
BRS-PLT	41.05 ± 0.13	33.33 ± 0.11	24.49 ± 0.09	25.15 ± 0.08	26.26 ± 0.09	23.33 ± 0.10	12.01 ± 0.03	26.10 ± 0.08	31.45 ± 0.11	24.92 ± 1.21
COV$_\beta$-PLT	43.52 ± 0.13	39.31 ± 0.13	36.12 ± 0.14	29.19 ± 0.08	32.14 ± 0.12	34.75 ± 0.15	14.35 ± 0.03	31.61 ± 0.09	41.66 ± 0.15	23.61 ± 1.36

to P@k:

$$1-\text{Cov@}k_{\mathcal{D}} = \frac{1}{n}\sum_{i=1}^{n} \text{Cov@}k(h(x^{(i)}), \boldsymbol{y}^{(i)}) \tag{26}$$

$$= \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{m} \boldsymbol{y}_j^{(i)} \cdot \mathbb{1}\left[j \in \text{top}_k(h(x^{(i)}))\right] . \tag{27}$$

Alternatively, we could tighten the condition required to count a label as "covered", by making the number of correct positive predictions dependent on the sample size. Introducing a threshold parameter $\alpha \in [0, 1]$, we can define α-coverage as

$$\alpha\text{Cov@}k = m^{-1}\sum_{j=1}^{m} \mathbb{1}\left[\text{R@}k \geq \alpha\right], \tag{28}$$

the fraction of labels for which a recall of at least α is achieved. This reduces to standard coverage when setting $\alpha = 0$ and to fixed sample-size coverage for $\alpha = 1/n'$.

6.2 Online Prediction

In our current approach, the predictions are sensitive to the order in which samples are presented. At first, the prediction procedure will select likely labels, but once the probability of these being

covered is estimated to be high, it has to switch to less and less likely (tail) labels as its prediction. As shown, this leads to significantly reduced precision scores. Ideally, we would like our prediction to depend only on the current instance x, and not on the prediction history. The n'-coverage might help in defining such a procedure, but further investigation into this problem is necessary.

6.3 Saturation of Performance

Looking forward, we think that addressing the problem of appropriate metrics will become more and more important as the performance on standard metrics starts to saturate. As long as the current state-of-the-art classifiers make a significant number of mistakes in their top-k predictions,[6] improving metrics like precision at k can be expected to also improve the suitability of these classifiers in their practical application. However, on certain datasets such as Wiki10-31K [Zubiaga 2012] and AmazonCat-13k [McAuley et al. 2015], precision at one has reached 89.45% and 96.77%, so further improvements have to come from selecting *which* of the predicted relevant labels to place in the top spots.

We hope that this paper will encourage future research and further transfer of ideas between recommendation systems and XMLC communities regarding suitable performance metrics for problems with large output spaces and long-tailed distributions.

REFERENCES

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

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

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

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

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

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

Keith Bradley and Barry Smyth. 2001. Improving recommendation diversity. In *Proceedings of the twelfth Irish conference on artificial intelligence and cognitive science, Maynooth, Ireland*, Vol. 85. Citeseer, 141–152.

Pablo Castells, Jun Wang, Rubén Lara, and Dell Zhang. 2011. Workshop on novelty and diversity in recommender systems - DiveRS 2011. In *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11*. ACM Press, Chicago, Illinois, USA, 393. http://dl.acm.org/citation.cfm?doid=2043932.2044019

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

Ed Coffman, M.R. Garey, and David Johnson. 1996. *Approximation Algorithms for NP-Hard Problems*. 46–93.

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

[6] Given that these datasets contain a significant amount of missing labels, we cannot expect the performance metrics to reach 100% even for a perfect classifier.

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

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

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

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

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

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

Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems – A survey. *Knowledge-Based Systems* 123 (May 2017), 154–162.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2783258.2783381

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

Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018a. 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.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018b. Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 993–1002.

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

Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*. 784–791.

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

Erik Schultheis, Marek Wydmuch, Rohit Babbar, and Krzysztof Dembczynski. 2022. On Missing Labels, Long-tails and Propensities in Extreme Multi-label Classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA*. https://doi.org/10.1145/3534678.3539466

Tong Wei and Yu-Feng Li. 2020. Does Tail Label Help for Large-Scale Multi-Label Learning? *IEEE Transactions on Neural Networks and Learning Systems* 31, 7 (2020), 2315–2324. https://doi.org/10.1109/TNNLS.2019.2935143

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

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

Marek Wydmuch, Kalina Jasinska-Kobus, Rohit Babbar, and Krzysztof Dembczynski. 2021. *Propensity-Scored Probabilistic Label Trees*. 2252–2256.

Hui Ye, Zhiyu Chen, Da-Han Wang, and Brian Davison. 2020. Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification. In *International Conference on Machine Learning*. PMLR, 10809–10819.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 5812–5822.

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

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

Arkaitz Zubiaga. 2012. Enhancing Navigation on Wikipedia with Social Tags. https://doi.org/10.48550/ARXIV.1202.5469

Algorithm 2 PLT - (Greedy) Predict Top Label for Cov@$k(T, H, x^n, k)$

$f_j \leftarrow 1$ for all $j \in [m]$ ▷ Initialize a vector f for calculation of a probability of a "failure"
for $x^{(i)} \in x^n$ **do** ▷ For all the examples in the dataset
  $\hat{\mathbf{z}}^{(i)} \leftarrow \mathbf{0}$, $f_{\max} \leftarrow \max_{j\in[m]} f_j$, $Q \leftarrow \emptyset$ ▷ Initialize solution vector $\hat{\mathbf{z}}^{(i)}$, $f_{\max}$ and a priority queue Q, ordered ascending by $\hat{c}(v, x^{(i)})$
  $g(r_T, x^{(i)}) \leftarrow -\log \hat{\eta}(x^{(i)}, r_T)$ ▷ Calculate cost $g(r_T, x^{(i)})$ for the tree
  $\hat{c}(r_T, x^{(i)}) \leftarrow g(r_T, x^{(i)}) + \log w_{\max} - \log \max_{j\in\mathcal{L}_{r_T}} g_j$ ▷ Calculate est. cost $\hat{c}(r_T, x^{(i)})$ for the tree root
  Q.add($(r_T, g(r_T, x^{(i)}), \hat{c}(r_T, x^{(i)}))$ ▷ Add the tree root with cost $g(r_T, x^{(i)})$ and est. $\hat{c}(r_T, x^{(i)})$ to the queue
  while $\|\hat{\mathbf{z}}^{(i)}\|_1 < k$ **do** ▷ While the number of predicted labels is less than k
    $(v, g(v, x^{(i)}), \_) \leftarrow Q$.pop() ▷ Pop the element with the lowest cost from the queue
    if v is a leaf **then** ▷ If the node is a leaf
      $j \leftarrow L_v$ ▷ For the corresponding label j with leaf node v
      $\hat{z}_j \leftarrow 1$, $f_j \leftarrow (1 - \hat{\eta}_j(x^{(i)}))f_j$ ▷ Set label j as predicted in solution vector $\hat{z}^{(i)}$ and update expected probability of "failure" f_j
    else for $v' \in \mathrm{Ch}(v)$ **do** ▷ If the node is an internal node, for all child nodes
      $g(v', x^{(i)}) \leftarrow g(v, x^{(i)}) - \log \hat{\eta}(x^{(i)}, v')$ ▷ Compute $g(v', x^{(i)})$ using $\hat{\eta}(v', x^{(i)}) \in H$
      $\hat{c}(v', x^{(i)}) \leftarrow g(v', x^{(i)}) + \log f_{\max} - \log \max_{j\in L_{v'}} g_j$ ▷ Calculate estimation $\hat{c}(v', x^{(i)})$
      Q.add($(v', g(v', x^{(i)}), \hat{c}(v', x^{(i)}))$) ▷ Add the node, cost $g(v', x^{(i)})$, and est. $\hat{c}(v', x^{(i)})$ to the queue

A PROBABILISTIC LABEL TRESS FOR COVERAGE@k

In this section we introduce probabilistic labels trees (PLTs) that allow for efficiently retrieving only top-k labels with highest conditional probabilities. Then we show how to apply our greedy-approach for coverage@k (Algorithm 1) to PLTs.

A.1 Probabilistic labels trees

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

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

$$\mathbb{P}[z_v = 1|\boldsymbol{x}] = \eta_v(\boldsymbol{x}) = \prod_{v'\in\mathrm{Path}(v)} \eta(\boldsymbol{x}, v')\,, \tag{29}$$

where $\eta(\boldsymbol{x}, v) = \mathbb{P}\left[z_v = 1|z_{\mathrm{pa}(v)} = 1, \boldsymbol{x}\right]$ for non-root nodes, and $\eta(\boldsymbol{x}, v) = \mathbb{P}[z_v = 1\,|\,\boldsymbol{x}]$ for the root. Notice that (29) can also be stated as recursion:

$$\eta_v(\boldsymbol{x}) = \eta(\boldsymbol{x}, v)\eta_{\mathrm{pa}(v)}(\boldsymbol{x})\,, \tag{30}$$

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

$$\eta_{l_j}(\boldsymbol{x}) = \eta_j(\boldsymbol{x})\,, \quad \text{for } l_j \in L_T\,. \tag{31}$$

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

A.2 Greedy algorithm for Cov@k for PLTs

Wydmuch et al. [2021] introduced A^*-search based algorithm for finding efficiently k leaves, with highest value of $w_j\hat{\eta}_j(\boldsymbol{x})$, where $w_j \in [0, \infty)$ is given weight for label j. We use the same procedure to find k leaves, with highest value of $g_j = f_j - (1 - \hat{\eta}_j(x^{(i)}))f_j = f_j\hat{\eta}_j(x^{(i)})$, using values f_j as weights for each instance. The outline of the search method presented below is an adapted description from [Wydmuch et al. 2021], and the whole algorithm is presented in Algorithm 2.

We introduce cost function $c(l_j, \boldsymbol{x})$ for each path from the root to a leaf. Notice that:

$$f_j\hat{\eta}_j(\boldsymbol{x}) = \exp\left(-\left(-\log q_j - \sum_{v\in\mathrm{Path}(l_j)} \log\hat{\eta}(\boldsymbol{x}, v)\right)\right). \tag{32}$$

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

$$c(l_j, \boldsymbol{x}) = \log f_{\max} - \log f_j - \sum_{v\in\mathrm{Path}(l_j)} \log\hat{\eta}(\boldsymbol{x}, v)\,, \tag{33}$$

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

$$g(v, \boldsymbol{x}) = - \sum_{v' \in \text{Path}(v)} \log \hat{\eta}(\boldsymbol{x}, v') \tag{34}$$

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

$$h(v, \boldsymbol{x}) = \log f_{\max} - \log \max_{j \in L_v} f_j \tag{35}$$

is a heuristic function estimating the cost of reaching the best leaf node from node v. The A^*-search in our procedure evaluates nodes in the ascending order of their estimated cost values $\hat{c}(l_j, \boldsymbol{x})$. To guarantee that A^*-search finds the optimal solution—top-k labels with the highest $f(l_j, \boldsymbol{x})$ and thereby top-k labels with the highest $f_j \hat{\eta}_j(\boldsymbol{x})$—we need to ensure that $h(v, \boldsymbol{x})$ is admissible, i.e., it never overestimates the cost of reaching a leaf node [Russell and Norvig 2009]. We also would like $h(v, \boldsymbol{x})$ to be consistent, making the A^*-search optimally efficient, i.e., there is no other algorithm used with the heuristic that expands fewer nodes [Russell and Norvig 2009]. The proof that the presented search algorithm is both admissible and consistent can be found in [Wydmuch et al. 2021].

Table 3: Mean performance with standard errors, rounded to two decimal places, of different variants of COV$_beta$-PLT on standard precision, propensity-scored precision, and coverage@$\{1, 3, 5\}$ [%] and test CPU time per instance [ms]. A and B indicate values of the parameters of the propensity model for estimating values of p_j. The best result for each measure is in bold.

	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	Cov@1	Cov@3	Cov@5
	EurLex-4K, $A = 0.55$, $B = 1.5$								
COV$_{\beta=0}$-PLT	62.83 ± 0.17	55.19 ± 0.11	48.12 ± 0.09	41.74 ± 0.11	45.29 ± 0.10	47.31 ± 0.13	**44.77 ± 0.12**	**61.05 ± 0.08**	**66.39 ± 0.17**
COV$_{\beta=0.25}$-PLT	75.64 ± 0.08	66.12 ± 0.13	55.84 ± 0.12	**42.06 ± 0.08**	**47.19 ± 0.11**	**49.76 ± 0.16**	39.84 ± 0.11	56.15 ± 0.15	62.95 ± 0.27
COV$_{\beta=0.5}$-PLT	79.22 ± 0.16	67.85 ± 0.10	56.75 ± 0.12	40.50 ± 0.09	46.35 ± 0.08	49.40 ± 0.14	35.20 ± 0.13	52.59 ± 0.08	60.87 ± 0.16
COV$_{\beta=1}$-PLT	**81.10 ± 0.09**	**68.50 ± 0.08**	**57.04 ± 0.14**	38.83 ± 0.08	45.32 ± 0.08	48.66 ± 0.15	30.44 ± 0.13	48.96 ± 0.07	58.21 ± 0.21
	AmazonCat-13K, $A = 0.55$, $B = 1.5$								
COV$_{\beta=0}$-PLT	56.57 ± 0.14	50.56 ± 0.09	42.85 ± 0.07	**57.46 ± 0.12**	58.66 ± 0.11	59.63 ± 0.10	**80.07 ± 0.07**	**86.51 ± 0.09**	**88.62 ± 0.10**
COV$_{\beta=0.25}$-PLT	92.28 ± 0.03	78.52 ± 0.05	63.93 ± 0.08	56.41 ± 0.03	**66.65 ± 0.05**	**72.93 ± 0.12**	64.67 ± 0.10	74.97 ± 0.08	79.30 ± 0.08
COV$_{\beta=0.5}$-PLT	92.94 ± 0.03	78.74 ± 0.04	64.00 ± 0.08	53.18 ± 0.02	65.03 ± 0.04	72.17 ± 0.11	52.98 ± 0.08	68.93 ± 0.08	76.30 ± 0.11
COV$_{\beta=1}$-PLT	**93.20 ± 0.02**	**78.82 ± 0.04**	**64.00 ± 0.08**	51.66 ± 0.02	64.12 ± 0.04	71.64 ± 0.10	40.02 ± 0.08	60.84 ± 0.07	72.58 ± 0.08
	Wiki10-31K, $A = 0.55$, $B = 1.5$								
COV$_{\beta=0}$-PLT	55.96 ± 0.16	47.05 ± 0.15	41.45 ± 0.11	**20.13 ± 0.07**	**18.31 ± 0.04**	**17.90 ± 0.06**	**9.79 ± 0.05**	**15.95 ± 0.02**	**19.29 ± 0.05**
COV$_{\beta=0.25}$-PLT	72.31 ± 0.11	65.12 ± 0.06	57.53 ± 0.12	18.35 ± 0.04	17.42 ± 0.04	17.47 ± 0.06	7.90 ± 0.04	12.66 ± 0.04	15.37 ± 0.06
COV$_{\beta=0.5}$-PLT	77.04 ± 0.14	69.07 ± 0.09	60.54 ± 0.11	17.36 ± 0.02	16.87 ± 0.02	17.04 ± 0.06	6.79 ± 0.01	11.08 ± 0.03	13.70 ± 0.07
COV$_{\beta=1}$-PLT	**80.42 ± 0.16**	**71.44 ± 0.09**	**62.32 ± 0.12**	16.24 ± 0.04	16.21 ± 0.03	16.55 ± 0.05	5.56 ± 0.03	9.47 ± 0.04	12.10 ± 0.05
	WikiLSHTC-325K, $A = 0.5$, $B = 0.4$								
COV$_{\beta=0}$-PLT	55.59 ± 0.30	36.18 ± 0.19	26.71 ± 0.15	**34.81 ± 0.19**	**36.50 ± 0.21**	38.73 ± 0.25	**35.33 ± 0.10**	**47.03 ± 0.14**	**51.69 ± 0.17**
COV$_{\beta=0.25}$-PLT	65.04 ± 0.25	**43.33 ± 0.18**	**31.91 ± 0.16**	31.65 ± 0.13	35.87 ± 0.18	**38.97 ± 0.22**	29.69 ± 0.05	42.43 ± 0.11	47.94 ± 0.13
COV$_{\beta=0.5}$-PLT	**65.17 ± 0.23**	43.23 ± 0.16	31.84 ± 0.15	29.64 ± 0.11	34.75 ± 0.16	38.04 ± 0.20	26.56 ± 0.04	40.33 ± 0.09	46.16 ± 0.12
COV$_{\beta=1}$-PLT	64.82 ± 0.21	42.91 ± 0.15	31.63 ± 0.14	28.07 ± 0.10	33.67 ± 0.14	37.07 ± 0.18	23.45 ± 0.03	38.14 ± 0.08	44.30 ± 0.10
	WikipediaLarge-500K, $A = 0.5$, $B = 0.4$								
COV$_{\beta=0}$-PLT	55.05 ± 0.26	39.29 ± 0.19	30.77 ± 0.15	**34.41 ± 0.17**	**35.11 ± 0.18**	36.91 ± 0.22	**31.52 ± 0.08**	**44.20 ± 0.13**	**49.55 ± 0.17**
COV$_{\beta=0.25}$-PLT	66.85 ± 0.28	48.32 ± 0.21	37.69 ± 0.18	31.70 ± 0.12	34.40 ± 0.15	**36.98 ± 0.19**	26.48 ± 0.05	39.43 ± 0.10	45.41 ± 0.14
COV$_{\beta=0.5}$-PLT	67.82 ± 0.26	**48.66 ± 0.20**	**37.87 ± 0.17**	29.91 ± 0.10	33.40 ± 0.14	36.16 ± 0.17	23.58 ± 0.04	37.19 ± 0.08	43.54 ± 0.12
COV$_{\beta=1}$-PLT	**68.01 ± 0.25**	48.63 ± 0.19	37.81 ± 0.16	28.43 ± 0.09	32.46 ± 0.12	35.33 ± 0.16	20.63 ± 0.03	34.86 ± 0.07	41.57 ± 0.10
	Amazon-670K, $A = 0.6$, $B = 2.6$								
COV$_{\beta=0}$-PLT	42.80 ± 0.13	38.75 ± 0.13	35.65 ± 0.14	**29.50 ± 0.09**	**32.39 ± 0.12**	**34.82 ± 0.15**	**14.56 ± 0.04**	**32.08 ± 0.10**	**42.02 ± 0.16**
COV$_{\beta=0.25}$-PLT	43.52 ± 0.13	39.31 ± 0.13	36.12 ± 0.14	29.19 ± 0.08	32.14 ± 0.12	34.75 ± 0.15	14.35 ± 0.03	31.61 ± 0.09	41.66 ± 0.15
COV$_{\beta=0.5}$-PLT	44.00 ± 0.13	39.64 ± 0.13	36.34 ± 0.15	28.83 ± 0.08	31.87 ± 0.11	34.63 ± 0.15	14.00 ± 0.03	31.02 ± 0.09	41.28 ± 0.15
COV$_{\beta=1}$-PLT	**44.49 ± 0.13**	**39.93 ± 0.13**	**36.51 ± 0.15**	28.35 ± 0.07	31.53 ± 0.11	34.45 ± 0.14	13.41 ± 0.03	30.09 ± 0.08	40.65 ± 0.14

B ADDITIONAL RESULTS FOR COV$_\beta$-PLT

In Table 3 we present the results of COV$_\beta$-PLT algorithm described in Section 5, with 4 different values of β.