ENHANCING TAIL PERFORMANCE IN EXTREME CLASSIFIERS BY LABEL VARIANCE REDUCTION

ABSTRACT

Extreme Classification (XC) architectures, which utilize a massive one-vs-all classifier layer at the output, have demonstrated remarkable performance on problems with large label sets. Nonetheless, these have also been observed to falter on tail labels with few representative samples. This phenomenon has been attributed to factors such as classifier over-fitting and missing label bias, and solutions involving regularization and loss re-calibration have been developed.

This paper explores the impact of label variance, a previously unexamined factor, on the tail performance in extreme classifiers. This paper presents a method to systematically reduce label variance in XC by effectively utilizing the capabilities of an additional, tail-robust teacher model.

Comprehensive experiments are conducted on a diverse set of XC datasets which demonstrate that LEVER can enhance tail performance by around 5% and 6% points in PSP and Coverage metrics respectively when integrated with leading extreme classifiers. Moreover, when added to the top-performing Renée classifier, it establishes a new state-of-the-art. Extensive ablations and analysis substantiate the efficacy of our design choices. Code and datasets will be released for research purposes.

1 INTRODUCTION

Extreme Classification (XC) addresses tasks where a data point needs to be mapped to the *subset* of its relevant labels from a large label space. Deep architectures, that comprise a neural network encoder followed by a massive one-vs-all classification layer at the output, have become the de-facto standard for contemporary XC algorithms and have demonstrated remarkable results on several large-scale applications (Agrawal et al., 2013; Yadav et al., 2021; Chang et al., 2020; Beygelzimer et al., 2009) . Nonetheless, such over-parameterized classification layers have also been observed to under-perform on labels with limited representative samples (Wei et al., 2021). As a result, such tail labels, which constitute a majority of the label space and provide niche and informative results in user-facing applications (Jain et al., 2016), are inaccurately classified, thus diminishing their utility.

The challenge of enhancing the performance of extreme classifiers on tail labels has been the focus of some recent studies. These investigations have identified multiple factors contributing to the hardness of tail labels and proposed solutions for alleviating them. One key idea has been to constrain the capacity of tail classifiers thereby mitigating the chances of over-fitting (Dahiya et al., 2021b). A separate line of work has dealt with the effects of false negatives, commonly known as missing labels, on tail performance and proposed simple loss re-calibration-based solutions (Qaraei et al., 2021).

This paper brings to light an additional, yet previously unexamined factor that leads to a decline in tail performance: label variance. In extreme classification, label variance refers to the inaccuracies introduced in the ground truth when a complex distribution of label relevance is approximated with discrete subsets of labels for each data point. For example, in the recommendation task of associating users with items they prefer, feedback in the form of user clicks is subject to variability as a user's interests can fluctuate over time. Consequently, click-based ground truth collected within a finite timeframe may not be entirely accurate. Label variance can also occur in datasets with expert annotation due to differences in relevance judgment among experts. Furthermore, the approximation of large-scale data tasks through a finite training sample set, necessitated by cost considerations, can introduce additional label variance. The presence of label variance results in an imprecise ground truth in the constructed dataset, thereby compromising the quality of the trained models. Tail labels, by their very definition of being infrequently sampled, are particularly vulnerable to this problem.

It's important to note that label variance is different from missing labels as it does not introduce any systematic biases in the ground truth.

This paper presents a solution to address label variance by utilizing the model distillation framework, as inspired by recent research (Menon et al., 2021). In scenarios where two probabilistic models with varying generalization capabilities exist, the more accurate model (the "teacher") can provide reliable relevance targets to enhance the performance of the less accurate one. This work extends the framework from (Menon et al., 2021) to extreme classification tasks using one-vs-all classifiers, providing a robust theoretical foundation. It introduces LEVER, a novel framework based on knowledge distillation, that significantly improves tail classifier generalization with formal bounds. Furthermore, it presents an effective instantiation of this framework using a specialized Siamese teacher model, which has been shown to provide reliable relevance targets and yield significant gains on tail labels.

Comprehensive experiments are conducted on a diverse set of XC datasets which demonstrate that LEVER can enhance tail performance by around 5% and 6% points in PSP and Coverage metrics respectively when integrated with leading extreme classifiers. Moreover, when added to the top-performing Renee classifier, it establishes a new state-of-the-art. Extensive ablations and analysis substantiate the efficacy of our design choices.

This paper makes the following key contributions: (1) A principled LEVER framework to mitigate the label variance effects on tail classifiers in XC; (2) An effective and well-calibrated Siamese-style model as a teacher with LEVER; (3) Extensive experimentation using multiple state-of-the-art approaches and diverse benchmarks that effectively demonstrate the utility and generality of the proposed approach; (4) Additionally, two new datasets are proposed which resemble real-world applications like query-keyword matching and query-autocompletion tasks.

2 Related Work

2.1 Extreme Classification

Recent advancements (Dahiya et al., 2021b; Chang et al., 2020; Kharbanda et al., 2022) in XC have leveraged deep network-based representations like LSTM (You et al., 2018), Transformer (Zhang et al., 2021; Jiang et al., 2021) or customized architectures (Dahiya et al., 2021b) to generate rich semantic representations of inputs. These are then assigned to appropriate labels via an OvA classifier layer. To facilitate efficient learning with large label sets, techniques such as multi-staged encoder refinement (Dahiya et al., 2021b; Zhang et al., 2021; Jiang et al., 2021), hierarchical label search, and hard-negative sampling (Dahiya et al., 2022; 2021b; Zhang et al., 2021; Jiang et al., 2021; Mittal et al., 2021a) have been introduced. Furthermore, simultaneous training of the deep encoder and OvA classifiers has been demonstrated to boost performance in leading XC approaches like ELIAS (Gupta et al., 2022), CascadeXML (Kharbanda et al., 2022) and Renée (Jain et al., 2023). However, despite these advancements, many of these approaches share a common limitation: a decline in performance for tail labels, which is the primary focus of this paper.

2.2 Enhancing Tail Performance in XC

Extreme Classifiers have been observed to under-perform on tail labels with limited representative samples. This phenomenon has been attributed to various factors, and several approaches have been proposed to address them.

Over-fitting of OvA Classifiers: OvA classifiers, which employ a distinct classifier for each label, are massively parameterized in scenarios with large label sets. Consequently, they are susceptible to overfitting on tail labels with scarce representative samples. To counteract this, various classifier regularization techniques have been introduced. For instance, ProXML (Babbar & Schölkopf, 2019) employs an L1-regularizer, and GLaS (Guo et al., 2019) uses a label-correlation based regularizer.

Bias due to Missing Labels: In Extreme Classification (XC) datasets, which are often too large for exhaustive labeling, missing or false negative labels are a frequent issue. These missing labels introduce systematic biases into the ground truth and are known to significantly impact tail labels. Strategies to address tail labels typically involve estimating the missing propensities for labels first

and then recalibrating the loss through simple weighting (Jain et al., 2016; Wei et al., 2021; Wydmuch et al., 2021). The phenomenon of missing label bias, while distinct, bears a close relationship to the issue of label variance.

Data Scarcity in Tail Labels: XC datasets frequently contain tail labels with a limited number of positive data samples. To mitigate this data scarcity, data augmentation techniques like TAUG (Wei et al., 2021) and Gandalf (Kharbanda et al., 2023) have been introduced. However, these methods lack formal guarantees and are seen to not perform consistently well across different datasets in this paper. Another line of work has exploited label-side features to improve the tail performance (Xiong et al., 2020; Dahiya et al., 2021a; 2022; Jain et al., 2023). Approaches such as NGAME (Dahiya et al., 2022) and Renée (Jain et al., 2023) leverage label correlations via a Siamese encoder, which semantically aligns similar labels in an embedding space, thereby facilitating information sharing among tail labels. However, these methods primarily focus on enhancing encoder robustness and do not explicitly address the quality of subsequent OvA classifiers. Our proposed model shares similarities with these approaches through its use of a Siamese teacher but distinguishes itself by learning a specialized teacher model suitable for distillation and focusing on improving OvA classifiers in a principled way.

In addition to these known issues, this paper introduces label variance as an additional, but important, consideration pertaining to tail performance in XC.

3 LEVER: LABEL VARIANCE REDUCTION IN EXTREME CLASSIFICATION

Label variance refers to the imprecision in the ground truth obtained through sampling, which is primarily due to approximation errors. These errors can negatively impact the performance of trained classifiers, particularly those on the tail. This section introduces LEVER, a principled framework based on knowledge distillation, designed to alleviate label variance and enhance the generalization capabilities of One-vs-All (OvA) classifiers. A practical and effective teacher model, based on a Siamese-style encoder, is also proposed.

3.1 PRELIMINARIES

An Extreme Classification (XC) task deals with a data point space $\mathcal{X}$, which is to be mapped onto a label space, represented as $\mathcal{Y} = \{0, 1\}^L$. Here, L signifies the number of labels, potentially reaching into the millions. The architecture of a deep extreme classification typically includes a deep encoder, $\mathcal{E}$, which generates a semantically rich representation, $\mathcal{E}(X)$, for any given input data point $X \in \mathcal{X}$. This process is succeeded by an extensive one-vs-all classifier layer $\{\mathbf{w}_l\}_{l=1}^L$. This layer assigns labels based on scores derived from $\mathbf{w}_l^\top \mathcal{E}(X)$, which are then sorted, and the highest scoring ones are selected as the relevant labels.

The model can be trained using various strategies, such as stagewise training, which trains the encoder and classifiers in separate stages, and end-to-end training, which trains both concurrently. This paper focuses on improving one-vs-all classifiers, so we adopt the stagewise approach where the encoder is fixed. Consequently, each label's classifier can be trained independently, simplifying theoretical analysis. Hereafter, the space of encoder embeddings, $\mathbf{x} \in \mathcal{E}(\mathcal{X}) \subset \mathbb{R}^D$ is referred to as our data point space.

For a data point $\mathbf{x}$, let $Y(\mathbf{x}) \in \{0, 1\}^L$ represent the set of relevant labels. In XC tasks, relevance is typically stochastic due to inherent variabilities in a user's preferences or annotators' judgments. Therefore, it is more appropriate to express relevance through a conditional probability distribution $\mathbb{P}(Y(\mathbf{x}) = \mathbf{y}|\mathbf{x}) \forall \mathbf{y} \in \{0, 1\}^L$. Note that this distribution sums up to 1 over all label subsets.

However, the full relevance distribution is seldom available as well as computationally expensive for model training. Consequently, it is a common practice to approximate the relevance distribution using a discrete sample of label subset $\mathbf{y} \sim \mathbb{P}(Y(\mathbf{x}) = \mathbf{y}|\mathbf{x})$. However, the sample might not be an accurate approximation of the whole distribution, and this imprecision is captured through variance in label relevance:

$$\mathbb{V}[\mathbf{y}] = \mathbb{E}_{Y|X}[\mathbf{y} - \mathbb{E}[\mathbf{y}]]^2$$
$$\mathbb{V}[y_l] = \mathbb{E}_{y_l|X}[y_l - \mathbb{E}[y_l]]^2 = \mathbb{P}(y_l = 1|\mathbf{x})(1 - \mathbb{P}(y_l = 1|\mathbf{x})) \tag{1}$$

The second expression signifies the variance in the marginal relevance of a label l to point $\mathbf{x}$, a term that is particularly useful in the analysis of one-vs-all classifiers. A larger variance indicates that a label set sampled randomly is considerably more imprecise.

To train a model, we initially construct a training set denoted as $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, where $\mathbf{x}_i \sim \mathcal{E}(\mathcal{X}), \mathbf{y}_i \sim \mathbb{P}(Y(\mathbf{x}_i) = \mathbf{y}_i|\mathbf{x}_i)$. Subsequently, an independent linear classifier is trained for each label l by solving a binary classification problem, using y_{il} as the target label for the ith data point. For clarity, we present the analysis for a single classifier, with the understanding that the same process applies to all classifiers. To avoid confusion, we omit the subscript l where it is not necessary.

A binary classification task involves minimizing the empirical risk of classification:

$$\hat{\mathbf{R}} = \min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, \mathbf{w}^\top \mathbf{x}_i) \tag{2}$$
$$\text{with, } \mathcal{L}(y, \mathbf{w}^\top \mathbf{x}) = Cyf(1, \mathbf{w}^\top \mathbf{x}) + (1 - y)f(0, \mathbf{w}^\top \mathbf{x}) \tag{3}$$

Here, f represents a convex classification surrogate such as hinge loss or logistic loss (Qaraei et al., 2021). Using a weight factor $C > 1$ is standard practice in class-imbalanced classification and mitigates the training bias in tail labels. For later use, we characterize such tail labels, with few positives as follows, where S is a pre-defined threshold.

$$\mathbb{E}_{\mathbf{x}}[p_x] \leq S \ll 1 \tag{4}$$
$$\text{where, } \quad p_x = \mathbb{P}(y = 1|\mathbf{x}) \tag{5}$$

In line with the standard practice (Kakade et al., 2008), we make certain assumptions. We assume that the norms of the weight vector $\mathbf{w}$ and the input vector $\mathbf{x}$ are bounded such that $\|\mathbf{w}\| \leq W$ and $\|\mathbf{x}\| \leq X$ respectively. Additionally, we presume that the function f exhibits Lipschitz continuity with a Lipschitz constant L.

The generalization performance of a trained classifier $\mathbf{w}^*$ is evaluated by its true population risk. A lower value of this risk indicates superior predictive capability:

$$\mathbf{R} = \mathbb{E}_{\mathbf{x}}[\mathcal{L}(y_{\mathbf{x}}, \mathbf{w}^{*\top} \mathbf{x})] \tag{6}$$

3.2 LEVER FRAMEWORK

The deviation between the empirical and true risks is a formal expression of a classifier's generalization performance. In our study, which focuses on data-dependent bounds based on variances, we adopt the approach outlined in (Maurer & Pontil, 2009). Applying Bennett's inequality, as suggested in the reference, with simplifications relevant for the problem at hand, provides us the following result:

Theorem 1. *Let $\mathcal{M}_N$ be the uniform covering number (Menon et al., 2021) corresponding to the classification loss $\mathcal{L}$. Then, given the definitions established earlier, For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over sampling the data points $\{\mathbf{x}\}_{i=1}^N$,*

$$\mathbf{R} \leq \hat{\mathbf{R}} + \mathcal{O}\Big(\sqrt{\mathbb{V}_{\mathbf{x}}\mathcal{L}(p_x, \mathbf{w}^\top \mathbf{x}) + (C+1)(LWX)^2 \cdot \mathbb{E}[\mathbb{V}_y[y|\mathbf{x}]]}\sqrt{\log(\mathcal{M}_N/\delta)/N} + \log(\mathcal{M}_N/\delta)/N\Big) \tag{7}$$

where, $\mathbb{V}_{\mathbf{x}}\mathcal{L}(p_x, \mathbf{w}^\top \mathbf{x}), \mathbb{V}_y[y|\mathbf{x}]$ are the variances in the loss function contributed by $\mathbf{x}$, and conditional variance of y respectively.

Proof is provided in the supplementary Section A

Theorem 1 establishes a direct correlation between the generalization performance of a classifier and the variance in labels $\mathbb{V}_y[y|\mathbf{x}]$. This implies that reducing this variance can enhance the effectiveness of the trained classifiers. Notably, if we have precise estimates of marginal relevance, denoted by $p_x = \mathbb{E}[y|\mathbf{x}]$, we can replace y with p_x, effectively reducing the variance term to 0 and thereby improving classifier generalization. This principle forms the foundation for the LEVER framework, which employs an additional teacher network to provide accurate estimates of p_x.

The label variance term in (7) can be further bounded, by using (4), as follows:

$$(C+1)(LWX)^2 \cdot \mathbb{E}[\mathbb{V}_y[y|\mathbf{x}]] \leq (C+1)(LWX)^2 \cdot \mathbb{E}[p_x] \leq (C+1)S(LWX)^2 \tag{8}$$

The terms $(C+1)$ and S are dependent on the label skew and inversely correlated with each other. The positive-negative balancing constant C depends on design choices. In the special case where there is complete balance, i.e., $C = \frac{1-S}{S}$, it is observed that the impact of label variance is confined within $(1-2S)(LWX)^2$. This indicates that labels in the tail, characterized by a smaller S, are more susceptible to variance effects compared to more prevalent labels. Therefore, it becomes evident that reducing label variance is crucial for enhancing the performance of tail classifiers.

3.3 A SIAMESE-STYLE TEACHER FOR LEVER

Recent studies have shown that Siamese Networks, when used as input encoders, exhibit impressive performance on tail labels (Dahiya et al., 2021a; 2022; Jain et al., 2023). This success can be attributed to the ability of Siamese encoders to learn correlations by utilizing label-side features. These features, often presented as descriptive text or structured graphs over labels, are commonly found in Extreme Classification (XC) applications. In fact, most recent XC datasets have started to incorporate them (Bhatia et al., 2016). Consequently, this allows for the sharing of information between semantically similar labels, effectively addressing the problem of data scarcity in tail labels. It's important to note, however, that a standalone Siamese model is insufficient as it tends to underfit data-rich head labels, thereby compromising overall prediction quality. This paper, therefore, proposes the use of Siamese Networks as teachers within the LEVER framework to enhance the tail performance of one-vs-all classifiers. By employing LEVER, we can improve the tail performance of one-vs-all classifiers without compromising their already excellent head accuracies.

A Siamese encoder is trained to map the features of data points, denoted as $\{\mathbf{x}_i\}_{i=1}^N$, and label features, represented as $\{\mathbf{z}_l\}_{l=1}^L$, into a common embedding space. The objective of this mapping is to ensure that labels relevant to a given data point are positioned closer in the embedding space, while those that are irrelevant are distanced. This is achieved by minimizing a triplet loss $[\mathbf{z}_k^\top \mathbf{x}_i - \mathbf{z}_l^\top \mathbf{x}_i + \Delta]_+$, where k and l are a negative and a positive label, respectively, for $\mathbf{x}_i$ and Δ is a margin enforced for better generalization (Dahiya et al., 2021a; 2022). The triplet loss is applied between all pairs of positive-negative labels for a given point. However, the triplet-loss is not probabilistically calibrated and does not provide reliable relevance targets for training a student.

To address this, we leverage a logistic-loss based objective that is found to be well-calibrated:

$$\min_{\substack{\text{Siamese} \\ \text{network}}} \sum_{l \in L} \sum_{\substack{k \in X_- \\ i \in X_+}} \log(1 + e^{\mathbf{z}_l^\top \mathbf{x}_k - \mathbf{z}_l^\top \mathbf{x}_i}) \tag{9}$$

The following Theorem demonstrates the calibration property of (9) assuming that the loss can be fully minimized, and therefore loss between each positive-negative pair is minimized.

Theorem 2. *Given a label* $\mathbf{z}$*, and a pair of data points* $\mathbf{x}_a, \mathbf{x}_b$*. Let* p_a, p_b *be the probabilities that the label is relevant to points* a, b *respectively. Then, assuming that (9) is fully minimized, the expected loss in (9) is minimized for* $p_a = 1/(1 + e^{-(m\mathbf{z}^\top \mathbf{x}_a + c)}), p_b = 1/(1 + e^{-(m\mathbf{z}^\top \mathbf{x}_b + c)})$

Proof is provided in supplementary Section A

The above result shows a direct connection between the Siamese model's scores and relevance probabilities which can be exploited as teacher targets. Values m, c can be treated as model hyperparameters and fitted by cross-validating.

4 CONTRIBUTED DATASETS

Motivation Performance evaluation of XML algorithms has largely relied on public benchmark datasets available from (Bhatia et al., 2016). In most of these datasets, the *intent* of a relevant label is largely available from its document's text. We refer to these as *single-intent* datasets. For example, in LF-AmazonTitles-131K, the product "clothing for men" might be associated with "formal shirts for men" or "casual shirts for men". In contrast, several real-world XML applications belong to a *multi-intent* setting where the label of a data point is often associated with diverse categories. For instance, in query auto-completion (Yadav et al., 2021) where a user's partial query needs to be completed, the query prefix and suffix only contain partial user intent, *e.g.* a prefix "face" could map to "book" or "wash". Such *multi-intent* datasets can be challenging for XML but are under-represented among existing benchmarks. To bridge this gap, this paper contributes two new *multi-intent* benchmarks.

Contributed datasets: Two new datasets, LF-AOL-270K and LF-WikiHierarchy-1M are curated. LF-AOL-270K involves the query auto-completion task of matching a query prefix with completing suffixes. It is curated from public AOL search logs (Pass et al., 2006). LF-WikiHierarchy-1M involves the taxonomy completion task Benaouicha et al. (2016) of matching a Wikipedia category to its parent categories (Zesch & Gurevych, 2007). As the parent typically generalizes the child category's intent, this aligns closely with the real-world application of query-to-keyword matching. Source data is processed by following steps provided in (Bhatia et al., 2016). Complete dataset creation details and dataset statistics are provided in B.2.

5 EXPERIMENTS AND RESULTS

Datasets: LEVER was evaluated on a diverse set of datasets, encompassing both full-text and short-text collections, as well as novel multi-intent datasets. Specifically, we utilized three full-text datasets (LF-Amazon-131K, LF-Wikipedia-500K, LF-WikiSeeAlso-320K), two short-text datasets (LF-AmazonTitles-131K, LF-AmazonTitles-1.3M), and two new multi-intent datasets (LF-WikiHierarchy-1M and LF-AOL-270K). For detailed dataset statistics, please refer to Table 3 in the supplementary material. Additionally, LEVER was evaluated on a large proprietary query-to-keyword matching dataset (QK-20M) with 20M labels.

Evaluation Metrics To assess the performance of all XC methods, standard evaluation metrics were used, namely precision@k (P@k, k= 1, 3, and 5) and its propensity-weighted variant PSP@k (with k=1, 3, and 5). Detailed definitions for these metrics can be found in (Bhatia et al., 2016). Additionally, following the recommendations in (Schultheis et al., 2022), we also include coverage@k (C@k) as an important metric to evaluate tail label performance.

Baselines We apply LEVER to multiple strong OvA-based baselines, including CascadeXML (Kharbanda et al., 2022), ELIAS (Gupta et al., 2022), and Renée (Jain et al., 2023), demonstrating its effectiveness. Furthermore, we conducted comparative evaluations with various tail-specialized techniques that can be seamlessly integrated with OvA classifiers without necessitating architectural modifications. This comparison included other regularization-based methods such as GLaS (Guo et al., 2019) and $L2$-regularization, as well as recent data augmentation methods like TAUG (Wei et al., 2021) and Gandalf (Kharbanda et al., 2023). We also illustrated the capability of LEVER to improve tail performance when coupled with propensity weighting methods like Re-rank (Wei et al., 2021). For comprehensive details on model hyper-parameters, please refer to Section D in the supplementary material.

LEVER Implementation Details As discussed in Section 3, LEVER incorporates label-label and label-document correlations based on a Siamese teacher's embeddings. Hyper-parameters for encoder training were taken from (Dahiya et al., 2022). Additionally, Siamese training was conducted over mini-batches of labels in order to give more importance to tail labels, unlike the document batching approach used in (Dahiya et al., 2022), which is biased toward the head. The LEVER augmented dataset $\mathcal{D}_{aug}$ is created by initially adding each label as a document point, resulting in a dataset comprising $N + L$ documents and L labels. Encoder embeddings are then utilized to pool label and document embeddings together. For each label l, the τ nearest points from the pool are selected and added as ground truth. Note that, this process results in additional label-label pairs and label-document pairs, which are added into the expanded matrix either by augmenting existing pairs in the $N \times L$ sub-matrix or by forming new pairs in the $L \times L$ sub-matrix.

Table 1: LEVER can be applied to improve any OvA-based approach. When used with leading OvA approaches LEVER consistently boosts tail performance across all benchmarks, increasing PSP on average by 5.3% while maintaining comparable precision (1.4% gain on average). Coverage metrics (reported in Table 5 in the supplementary material) show similar trends with an average gain of 6.5%.

Model	LF-AmazonTitles-131K						LF-Amazon-131K					
	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
ELIAS	37.28	25.18	18.14	28.95	34.45	39.08	43.03	29.27	21.20	33.49	40.80	46.76
ELIAS + LEVER	**42.86**	**28.37**	**20.16**	**36.30**	**41.05**	**45.43**	**47.38**	**32.24**	**23.22**	**38.97**	**46.74**	**52.79**
CascadeXML	35.96	24.77	18.15	26.22	33.06	38.72	43.76	29.75	21.63	34.05	41.69	48.09
CascadeXML + LEVER	**43.16**	**28.66**	**20.59**	**35.86**	**41.65**	**46.86**	**48.24**	**32.82**	**23.73**	**39.09**	**47.55**	**54.18**
Renée	46.05	30.81	**22.04**	38.47	44.87	**50.33**	48.05	32.33	23.26	39.32	47.10	53.51
Renée + LEVER	**46.44**	**30.83**	21.92	**39.70**	**45.44**	50.31	**49.19**	**33.30**	**24.04**	**40.64**	**48.48**	**54.87**

	LF-Wikipedia-500K						LF-AmazonTitles-1.3M					
	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
ELIAS	81.94	62.71	48.75	33.58	43.92	48.67	47.48	42.21	38.60	18.79	23.20	26.06
ELIAS + LEVER	**82.44**	**63.88**	**50.03**	**36.94**	**49.28**	**55.03**	**48.91**	**43.17**	**39.28**	**23.68**	**27.43**	**29.72**
CascadeXML	77.00	58.30	45.10	31.25	39.35	43.29	47.14	41.43	37.73	15.92	20.23	23.16
CascadeXML + LEVER	**80.10**	**60.41**	**46.44**	**36.79**	**46.65**	**50.99**	**47.98**	**42.02**	**38.12**	**20.06**	**24.51**	**27.28**
Renée	84.95	66.25	51.68	37.10	50.27	55.68	**56.10**	**49.91**	**45.32**	28.56	33.38	36.14
Renée + LEVER	**85.02**	**66.42**	**52.05**	**42.50**	**54.86**	**60.20**	56.01	49.43	44.85	**33.55**	**36.82**	**38.81**

	LF-AOL-270K						LF-WikiHierarchy-1M					
	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
ELIAS	40.83	22.33	14.91	13.29	21.46	25.22	**95.27**	**94.25**	**92.45**	17.15	24.41	30.01
ELIAS + LEVER	**48.91**	**43.17**	**39.28**	**23.68**	**27.43**	**29.72**	94.02	91.97	89.50	**28.27**	**36.80**	**42.13**
CascadeXML	**41.20**	**22.12**	14.82	**12.58**	19.53	23.19	**94.88**	**93.69**	**91.79**	16.03	22.87	28.17
CascadeXML + LEVER	39.41	21.78	**14.99**	11.96	**21.30**	**27.59**	94.77	93.54	91.56	**20.14**	**27.49**	**33.01**
Renée	40.97	23.34	15.85	15.06	26.36	31.97	95.01	**93.99**	**92.24**	19.69	27.36	33.20
Renée + LEVER	**41.70**	**24.76**	**17.07**	**20.38**	**37.07**	**45.13**	**95.19**	93.90	92.07	**24.79**	**32.74**	**38.29**

Performance on SOTA OvA methods Table 1 demonstrates LEVER's effectiveness when applied to leading classifier-based XC methods, including CascadeXML, ELIAS, and Renée, along with their LEVER-based counterparts. LEVER consistently improves P@1 and PSP@1 on average by 2% and 5%, respectively, across all models and datasets. When applied to Renée, LEVER achieves new state-of-the-art performance, increasing PSP@1 by up to 5% while maintaining comparable precision. Notably, LEVER proves highly effective on smaller datasets (LF-AmazonTitles-131K, LF-Amazon-131K), highlighting its importance when training data is limited. Table 13 in Supplementary further illustrates LEVER's performance gains on a proprietary dataset containing 20M labels. Larger improvements in ELIAS and CascadeXML are attributed to these models not explicitly utilizing label features during training or initialization. In contrast, Renée, which uses the NGAME encoder for initialization, shows comparatively modest gains with LEVER. Moreover, Table 4 in the supplementary illustrates the performance of LEVER when combined with XReg (Prabhu et al., 2020), an extension of Parabel, showcasing that LEVER can effectively combine with non-DNN-based methods.

Comparison with Tail Extreme Classification Methods: In Table 2, we present a comparative analysis of Renée+LEVER against leading tail label-specialized methods. Note that these approaches can be easily integrated with OvA classifiers without any architectural modifications. These methods can be broadly categorized into two classes: (1) regularization-based techniques, such as GLaS (Guo et al., 2019) and $L2$-regularization. GLaS promotes the proximity of classifiers for labels with similar ground truth, while $L2$-regularization introduces an additional $L2$ loss between tail expert label embeddings and label classifiers. (2) Augmentation-based methods, such as TAUG (Wei et al., 2021) and Gandalf (Kharbanda et al., 2023), which introduce additional training data for labels. Detailed comparisons with other prominent Extreme Classification methods, including XR-Transformer (Zhang et al., 2021), ELIAS (Gupta et al., 2022), CascadeXML (Kharbanda et al., 2022), NGAME (Dahiya et al., 2022), and ECLARE (Mittal et al., 2021b), are provided in

Table 2: Comparison of LEVER with other tail specific XC approaches. LEVER outperforms regularization and augmentation-based methods by an average of 4% in coverage and 3% in PSP.

	LF-AmazonTitles-131K						**LF-AOL-270K**					
	C@1	C@3	C@5	PSP@1	PSP@3	PSP@5	C@1	C@3	C@5	PSP@1	PSP@3	PSP@5
Renée	31.31	53.50	61.03	38.47	44.87	50.33	12.40	29.77	36.53	15.06	26.36	31.97
Renée +TAUG	29.47	51.52	58.68	36.49	42.83	47.85	12.46	29.26	35.88	15.72	26.74	32.35
Renée + BoW	30.03	51.78	59.17	36.96	42.86	48.09	12.67	34.32	43.45	15.58	30.28	37.90
Renée + L2Reg	31.66	53.65	60.80	38.74	44.53	49.49	8.67	21.07	26.27	12.21	20.09	24.36
Renée + GLaS	31.90	54.02	61.15	38.74	44.53	49.49	12.36	29.41	36.06	14.67	26.11	36.75
Renée + Gandalf	**33.17**	**55.36**	**62.22**	**40.49**	**45.83**	**50.96**	12.63	29.82	36.31	15.10	26.64	32.17
Renée + LEVER	32.82	55.11	61.94	39.70	45.44	50.31	**17.43**	**42.54**	**52.01**	**20.38**	**37.07**	**45.14**
	LF-Wikipedia-500K						**LF-WikiHierarchy-1M**					
Renée	22.90	50.08	61.59	41.25	52.57	57.04	6.62	11.39	14.56	19.69	27.36	33.20
Renée + TAUG	19.88	44.74	56.13	33.76	46.54	52.16	3.59	7.19	9.94	16.95	24.06	29.69
Renée + BoW	22.92	49.64	61.40	36.66	49.79	55.55	7.84	14.77	18.39	24.25	31.10	36.30
Renée + L2Reg	26.52	53.95	65.14	39.55	52.42	57.43	5.55	9.89	12.91	18.44	25.79	31.37
Renée + GLaS	23.43	52.02	63.90	37.27	51.54	57.15	6.89	11.82	15.08	19.36	26.89	32.62
Renée + Gandalf	23.09	49.87	61.24	37.05	49.94	55.31	6.92	13.17	17.52	21.84	30.05	36.09
Renée + LEVER	**29.46**	**58.53**	**70.29**	**42.50**	**54.86**	**60.20**	**9.08**	**16.12**	**20.02**	**24.79**	**32.74**	**38.29**

Table 6 within the supplementary material. Our primary focus here is on tail label performance, hence we report PSP and coverage metrics. For a comprehensive view of all metrics, we direct the reader to the supplementary material.

LEVER consistently outperforms the second-best method by an average margin of 4% in coverage and 3% in PSP. Notably, on datasets characterized by significant skew and multi-intent scenarios, LEVER exhibits substantial performance gains in comparison to approaches like GLaS and Gandalf, which rely on ground truth data for modeling label correlations (please refer to Table 3 in the supplementary material for skew statistics for all datasets). Highly skewed datasets such as LF-WikiHierarchy-1M and LF-AOL-270K pose a significant challenge due to the scarcity of ground truth data for tail labels. With sparse ground truth annotations for tail labels, the pool of samples available for augmentation or regularization becomes substantially reduced. This challenge is further compounded by the multi-intent nature of the dataset, where labels may be associated with co-occurring labels of divergent intents. For example, in the query completion task on the AOL dataset, the label *"who wrote To Kill a Mockingbird"* may co-occur with labels like *"wholesale t-shirts"* or *"who am I"* as they share the prefix *"who"*. Training classifiers with such diverse targets can lead to associations between dissimilar labels, hampering classifier training. Using Bag of Words (BoW) features from label text to model label connections alleviates the multi-intent and skew issue to some extent as we observe Renée + BoW performs better than Renée + GLaS/Gandalf in LF-AOL-270K and LF-WikiHierarhy-1M. However, LEVER goes further by learning semantic associations between labels and documents through a tail-expert Siamese network, surpassing raw text-based methods.

Comparison with Siamese Teacher: In Table 9, we evaluate LEVER against its corresponding Siamese Teacher (label sided NGAME). LEVER utilizes the Siamese teacher to improve OvA performance for tail labels without affecting precision@k. Note in Table 9, the numbers correspond to NGAME encoder and not NGAME fusion, which is reported in (Dahiya et al., 2022) and (Bhatia et al., 2016).

Comparison with an ensemble of OvA classifier and tail-expert: To combine the strengths of OvA and encoder, another option might be to consider an ensemble model that uses predictions from the OvA model for head labels and the encoder predictions for the tail labels. Table 7 compares LEVER with an ensemble of OvA (Renée) and encoder (label sided NGAME). LEVER outperforms the ensemble on both precision and tail metrics. A more detailed discussion of this is provided in Section C.3 of the supplementary.

Choice of expert encoder: LEVER utilizes a 6-layer DistilBert as an expert encoder. In Table 11 in the supplementary we show results for two other light-weight encoders: a 3-layer MiniLM (Wang et al., 2020) and Astec Encoder (Dahiya et al., 2021b). We add the same number of neighbors for each label across all experts. We observe that a superior expert encoder leads to improved performance in both P and PSP.

Effect of sampling strategy: As discussed in the implementation details, for LEVER, the Siamese encoder is trained over mini-batches of labels rather than documents to give more importance to tail labels. Table 12 in supplementary compares the results of document mini-batch training to label mini-batch training. Label mini-batch training improves PSP on average by 1.4%.

Effect of varying τ: Figure 4 in the supplementary shows the effect of varying τ on LEVER's performance. Increasing τ leads to better performance on tail labels, while it hurts the head or torso label.

LEVER Computational Cost: Since LEVER is a training time-only modification, it leaves the inference costs unchanged while increasing the training time by at most 2x. Table 19 in the supplementary shows the training time for different models and datasets when combined with LEVER.

6 CONCLUSIONS

This paper presents a novel approach to address the challenges of tail performance in Extreme Classification (XC) by focusing on label variance, a previously unexplored factor. The proposed method, LEVER, leverages a tail-robust teacher model to systematically reduce label variance, thereby enhancing the performance of one-vs-all classifiers. The paper introduces an effective instantiation of this framework using a specialized Siamese teacher model. Experimental results on various XC datasets demonstrate significant improvements in tail performance metrics when LEVER is integrated with leading extreme classifiers and advances the state-of-the-art. This paper also released two new benchmarking datasets.

REFERENCES

Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 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*, pp. 13–24, 2013.

Rohit Babbar and Bernhard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8):1329–1351, 2019.

Mohamed Benaouicha, Mohamed Ali Hadj Taieb, and Malek Ezzeddine. Derivation of "is a" taxonomy from wikipedia category graph. *Eng. Appl. Artif. Intell.*, 50:265–286, 2016.

Alina Beygelzimer, John Langford, Yuri Lifshits, Gregory Sorkin, and Alex Strehl. Conditional probability tree estimation analysis and algorithms. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 51–58, 2009.

Kush Bhatia, Kunal Dahiya, Himanshu Jain, Purushottam Kar, Anshul Mittal, Yashoteja Prabhu, and Manik Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL `http://manikvarma.org/downloads/XC/XMLRepository.html`.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3163–3171, 2020.

Kunal Dahiya, Ananye Agarwal, Deepak Saini, K Gururaj, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International Conference on Machine Learning*, pp. 2330–2340. PMLR, 2021a.

Kunal Dahiya, Deepak Saini, Anshul Mittal, Ankush Shaw, Kushal Dave, Akshay Soni, Himanshu Jain, Sumeet Agarwal, and Manik Varma. Deepxml: A deep extreme multi-label learning framework applied to short text documents. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 31–39, 2021b.

Kunal Dahiya, Nilesh Gupta, Deepak Saini, Akshay Soni, Yajun Wang, Kushal Dave, Jian Jiao, Prasenjit Dey, Amit Singh, Deepesh Hada, et al. Ngame: Negative mining-aware mini-batching for extreme classification. *arXiv preprint arXiv:2207.04452*, 2022.

Chuan Guo, Ali Mousavi, Xiang Wu, Daniel N Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. Breaking the glass ceiling for embedding-based classifiers for large output spaces. *Advances in Neural Information Processing Systems*, 32, 2019.

Nilesh Gupta, Patrick Chen, Hsiang-Fu Yu, Cho-Jui Hsieh, and Inderjit Dhillon. Elias: End-to-end learning to index and search in large output spaces. *Advances in Neural Information Processing Systems*, 35:19798–19809, 2022.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 935–944, 2016.

Vidit Jain, Jatin Prakash, Deepak Saini, Jian Jiao, Ramachandran Ramjee, and Manik Varma. Renee: End-to-end training of extreme classification models. In *Proceedings of Machine Learning and Systems*, pp. To appear, 2023.

Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7987–7994, 2021.

Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/file/5b69b9cb83065d403869739ae7f0995e-Paper.pdf.

Siddhant Kharbanda, Atmadeep Banerjee, Erik Schultheis, and Rohit Babbar. Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. *arXiv preprint arXiv:2211.00640*, 2022.

Siddhant Kharbanda, Devaansh Gupta, Erik Schultheis, Atmadeep Banerjee, Vikas Verma, and Rohit Babbar. Gandalf : Data augmentation is all you need for extreme classification, 2023. URL https://openreview.net/forum?id=05ff9BRSMzE.

Gyuwan Kim. Subword language model for query auto-completion. *arXiv preprint arXiv:1909.00599*, 2019.

Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, Seungyeon Kim, and Sanjiv Kumar. A statistical perspective on distillation. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Bhaskar Mitra and Nick Craswell. Query auto-completion for rare prefixes. *CIKM '15: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015.

Anshul Mittal, Kunal Dahiya, Sheshansh Agrawal, Deepak Saini, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Decaf: Deep extreme classification with label features. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 49–57, 2021a.

Anshul Mittal, Noveen Sachdeva, Sheshansh Agrawal, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Eclare: Extreme classification with label graph correlations. In *Proceedings of the Web Conference 2021*, pp. 3721–3732, 2021b.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*, 2006.

Yashoteja Prabhu, Aditya Kusupati, Nilesh Gupta, and Manik Varma. Extreme regression for dynamic search advertising. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, pp. 456–464, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368223. doi: 10.1145/3336191.3371768. URL https://doi.org/10.1145/3336191.3371768.

Mohammadreza Qaraei, Erik Schultheis, Priyanshu Gupta, and Rohit Babbar. Convex surrogates for unbiased loss functions in extreme classification with missing labels. In *Proceedings of the Web Conference*, pp. 3711–3720, 2021.

Erik Schultheis, Marek Wydmuch, Rohit Babbar, and Krzysztof Dembczynski. 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*, pp. 1547–1557, 2022.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

Tong Wei, Wei-Wei Tu, Yu-Feng Li, and Guo-Ping Yang. Towards robust prediction on tail labels. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1812–1820, 2021.

Marek Wydmuch, Kalina Jasinska-Kobus, Rohit Babbar, and Krzysztof Dembczynski. Propensity-scored probabilistic label trees. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2252–2256, 2021.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2020.

Nishant Yadav, Rajat Sen, Daniel N Hill, Arya Mazumdar, and Inderjit S Dhillon. Session-aware query auto-completion using extreme multi-label ranking. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3835–3844, 2021.

Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. *arXiv preprint arXiv:1811.01727*, 137:138–187, 2018.

Torsten Zesch and Iryna Gurevych. Analysis of the Wikipedia category graph for NLP applications. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pp. 1–8, Rochester, NY, USA, 2007. Association for Computational Linguistics. URL https://aclanthology.org/W07-0201.

Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34:7267–7280, 2021.

A THEORETICAL PROOFS

Theorem 1. Let $\mathcal{M}_N$ be the uniform covering number (Menon et al., 2021) corresponding to the classification loss $\mathcal{L}$. Then, given the definitions established earlier, For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over sampling the data points $\{\mathbf{x}\}_{i=1}^{N}$,

$$\mathbf{R} \leq \hat{\mathbf{R}} + \mathcal{O}\Big(\sqrt{\mathbb{V}_{\mathbf{x}}\mathcal{L}(p_x, \mathbf{w}^\top \mathbf{x}) + (C+1)(LWX)^2 \cdot \mathbb{E}[\mathbb{V}_y[y|\mathbf{x}]]}\sqrt{\log(\mathcal{M}_N/\delta)/N} + \log(\mathcal{M}_N/\delta)/N\Big) \tag{10}$$

where, $\mathbb{V}_{\mathbf{x}}\mathcal{L}(p_x, \mathbf{w}^\top \mathbf{x}), \mathbb{V}_y[y|\mathbf{x}]$ are the variances in the loss function contributed by $\mathbf{x}$, and conditional variance of y respectively.

Proof. Applying Proposition 2. from (Menon et al., 2021) to our setting gives the following result:

$$\mathbf{R} \leq \hat{\mathbf{R}} + \mathcal{O}\Big(\sqrt{\mathbb{V}_{\mathbf{x},y}\mathcal{L}(y, \mathbf{w}^\top \mathbf{x})}\sqrt{\log(\mathcal{M}_N/\delta)/N} + \log(\mathcal{M}_N/\delta)/N\Big)$$

The following simplifications can be made from basic probabilistic calculus:

$$\begin{aligned}
\mathbb{V}_{\mathbf{x},y}[\mathcal{L}(y, \mathbf{w}^\top \mathbf{x})] &= \mathbb{V}_{\mathbf{x}}[\mathbb{E}_y[\mathcal{L}(y, \mathbf{w}^\top \mathbf{x})|\mathbf{x}]] + \mathbb{E}_{\mathbf{x}}[\mathbb{V}_y[\mathcal{L}(y, \mathbf{w}^\top \mathbf{x})|\mathbf{x}]] \\
&= \mathbb{V}_{\mathbf{x}}[\mathcal{L}(p_x, \mathbf{w}^\top \mathbf{x})] + \mathbb{E}[\mathbb{V}_y[y|\mathbf{x}](Cf^2(1, \mathbf{w}^\top \mathbf{x}) - f^2(0, \mathbf{w}^\top \mathbf{x}))] \\
&\leq \mathbb{V}_{\mathbf{x}}[\mathcal{L}(p_x, \mathbf{w}^\top \mathbf{x})] + \mathbb{E}[\mathbb{V}_y[y|\mathbf{x}](C+1)(LWX)^2]
\end{aligned} \tag{11}$$

where the last expression is derived by utilizing the Lipschitz continuity condition over f.

□

Theorem 2. Given a label $\mathbf{z}$, and a pair of data points $\mathbf{x}_a, \mathbf{x}_b$. Let p_a, p_b be the probabilities that the label is relevant to points a, b respectively. Then, assuming that (9) is fully minimized, the expected loss in (9) is minimized for $p_a = 1/(1 + e^{-(m\mathbf{z}^\top \mathbf{x}_a + c)}, p_b = 1/(1 + e^{-(m\mathbf{z}^\top \mathbf{x}_b + c)}$

Proof. The expected loss between the triplet is given by:

$$p_a(1 - p_b)\log(1 + e^{\mathbf{z}^\top \mathbf{x}_b - \mathbf{z}^\top \mathbf{x}_a}) + p_b(1 - p_a)\log(1 + e^{\mathbf{z}^\top \mathbf{x}_a - \mathbf{z}^\top \mathbf{x}_b}) \tag{12}$$

By setting the gradient of the above equation to 0, and then substituting the equations for p_a, p_b, the result is readily available □

B DATASET DETAILS

B.1 DATASET STATISTICS

Table 3 shows the statistics of benchmark datasets including the newly contributed *multi-intent* datasets.

B.2 MULTI-INTENT DATASET PREPARATION

B.2.1 LF-AOL-270K

Task Description: Query auto-completion involves matching a query prefix to completing suffixes, *e.g.* given a prefix, 'cheap nike s' recommending suffix completions like 'shoes', 'shirts' etc. LF-AOL-270K is curated from AOL search logs (Pass et al., 2006) for the task of query auto-completion where (prefix, suffix) pairs are modeled as (doc, label) pairs. Retrieved suffixes from this task can be combined with user prefixes to get full query completions as proposed in (Mitra & Craswell, 2015).

Table 3: Dataset Statistics. Pos-80% is an imbalance metric (Schultheis et al., 2022) defined as minimum fraction of class labels that retain 80% of all positive labels in the dataset. Lower value corresponds to higher skew.

	Dataset	Train Docs	Test Docs	Labels	Avg. Labels/Doc	Avg. Docs/Label	Pos-80%
Existing	LF-AmazonTitles-131K	294,805	134,835	131,073	2.29	5.15	47.5
	LF-Amazon-131K	294,805	134,835	131,073	2.29	5.15	47.5
	LF-WikiSeeAlso-320K	693,082	177,515	312,330	2.11	4.68	37.4
	LF-Wikipedia-500K	1,813,391	783,743	501,070	4.77	24.75	25.1
	LF-AmazonTitles-1.3M	2,248,619	970,237	1,305,265	22.20	38.24	28.9
New	LF-AOL-270K	3,922,479	519,352	272,825	2.01	28.83	11.6
	LF-WikiHierarchy-1M	1,589,378	397,952	976,214	25.98	42.31	7.3

Dataset generation: The dataset generation process involved three steps (i) Pre-processing, (ii) Prefix-suffix generation, and (iii) Post-processing

Pre-processing: Queries in AOL search logs were de-duplicated and non-alphanumeric characters were removed. Queries with less than three characters were filtered since auto-completion is rarely required for those. Additionally, steps prescribed in (Kim, 2019) were followed for pre-processing and train-test splits creation.

Prefix-suffix generation: After pre-processing, a shortlist of the top 10M popular suffixes was derived from the train split, based on their frequency in queries. These suffixes are popular n-grams (word-level) up to 100 characters appearing at the end of queries. Sampling was done to ensure that each train query has at least one suffix from 10M suffix shortlist. Ground truth suffixes were added for sampled prefixes yielding 9.3M suffixes and 5.67M distinct prefixes (training points). Using the 10M suffix shortlist, the process of sampling prefixes was repeated in the test split, resulting in 460K suffixes.

Post-processing: Train-test leakage was avoided by removing all prefixes in the test set that appeared in the train set. The suffix (label) set is derived from the intersection of train and test suffixes to have a fixed label set. Finally, the dataset contains 272K labels (suffixes), 3.9M training points (prefixes), and 519K test points (test prefixes).

Code to create the dataset from raw AOL search logs (Pass et al., 2006) will be released upon acceptance of this paper.

B.2.2 LF-WIKIHIERARCHY-1M

Task description: Taxonomy completion task involves matching a category with its generalized parent categories. LF-WikiHierarchy-1M uses Wikipedia categories to build a taxonomy completion task where documents are categories and labels are its parent categories. Articles in Wikipedia are assigned categories, which serve as semantic tags. (*e.g.* 'FIFA World Cup 2022' article has a category tag of 'Football'). These categories are arranged in a taxonomy-like structure where each category is linked to zero or more parent categories. The parent of a category is its direct generalization, *e.g.* the category 'Football' has direct parent categories 'Athletic Sports', 'Team sports' and 'Ball Games'. This taxonomy-like structure is called *Wikipedia Category graph (WCG)* and has been well studied in (Zesch & Gurevych, 2007; Benaouicha et al., 2016).

Dataset generation The dataset generation process involved four steps (i) Raw data collection, (ii) Pre-processing, (iii) Label set generation from WCG and (iv) Post-processing.

Raw data collection: The WCG is created by using the English Wikimedia dump as of 03/23 [1]. The dump contains the list of all Wikipedia categories and their links.

Pre-processing: To create the WCG we first filter out all meta categories used for Wikipedia maintenance, *e.g.* 'Wikipedia missing topics', 'Wikipedia new articles', 'Categories for renaming' etc. The complete list of filtered meta-categories will be released as part of the code. Post filtering, the resulting WCG is a directed acyclic graph with 1,993,526 categories (nodes) and 5,781,016 (edges). Each edge is a document-label pair.

[1] https://dumps.wikimedia.org/enwiki/20230301/

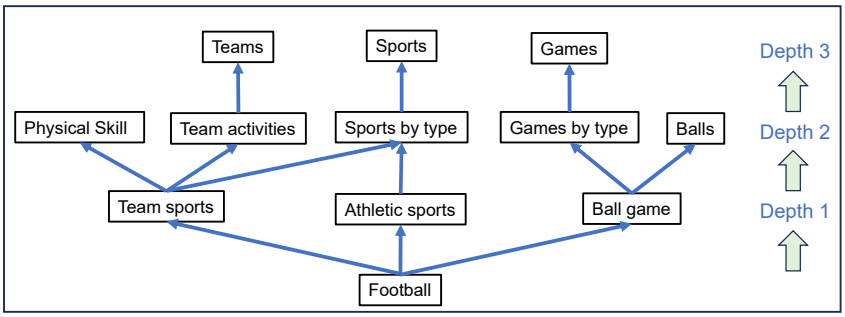

Figure 1: Snapshot of WCG graph starting from category 'Football'. All categories (nodes) reachable from 'Football' till the depth of 3 are added to its ground truth label set.

Label set generation from WCG: The WCG in its current form only contains direct parents and misses out on potentially important ground truth information. For example, the category 'Football' will not have categories like 'Sports', 'Athletic Sports', and 'Team activities' as its labels as they are not its *direct* parents. On the other hand, adding all reachable nodes as labels leads to vague document-label pairs. For example, starting from 'Football' one can reach the category 'Cosmopolitan mammals' as follows: 'Football' $\rightarrow$ 'Athletic sports' $\rightarrow$ 'Sports by type' $\rightarrow$ 'Sports' $\rightarrow$ 'Entertainment' $\rightarrow$ 'Human activities' $\rightarrow$ 'Humans' $\rightarrow$ 'Cosmopolitan mammals'.

To maximize the relevant ground truth document-label pairs while also avoiding wrong matches like *'Football'* $\rightarrow$ *'Cosmopolitan mammals'* we limit the traversal to a maximum depth of 3 which gave the optimal trade-off (i.e. maximize true positives while avoiding false positives). Thus, in the above example, only categories up to 'Sports' are added as labels. Please refer to Fig. 1 for more clarity. Subsequently, we get 1,987,330 documents and 976,214 labels with 51,643,812 edges between them. Note that both the number of documents and labels are less than the total number of categories (1,993,526). Some categories will not have a parent and therefore won't be added as a document. Similarly, categories that are not parents of any category will not be added as labels. The final dataset is created by taking an 80%-20% random train-test split.

Post-processing: The dataset contains categories that occur as both documents and labels. For example, the category 'Football' occurs both as a label (for 'Football clubs', 'History of Football', etc), and as a document. Since a category will never have itself as a label we filter off pairs like 'Football'$\rightarrow$'Football' during evaluation so as to not unfairly penalize Siamese-based models that rank such pairs at the top.

The processed dataset as well as the code to re-create it from Wikimedia dump will be released upon acceptance of this paper.

C Additional Results

C.1 LEVER's performance on non-DNN methods

Table 4 illustrates the performance of LEVER when combined with XReg (Prabhu et al., 2020), an extension of Parabel, showcasing that LEVER can effectively combine with non-DNN-based methods.

Table 4: Performance Comparison of XReg and XReg + LEVER on LF-AOL-270K and LF-AmazonTitles-131K

Dataset	Model	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	C@1	C@3	C@5
LF-AmazonTitles-131K	XReg	33.1	22.3	16.0	24.5	29.4	33.54	20.22	36.63	42.84
	XReg + LEVER	**38.0**	**24.7**	**17.6**	**31.4**	**35.1**	**39.1**	**25.84**	**43.47**	**49.68**
LF-AOL-270K	XReg	**27.0**	14.3	9.9	7.0	11.0	14.1	4.18	10.58	14.44
	XReg + LEVER	26.1	14.3	**10.1**	**9.2**	**17.9**	**24.0**	**6.79**	**20.59**	**28.65**

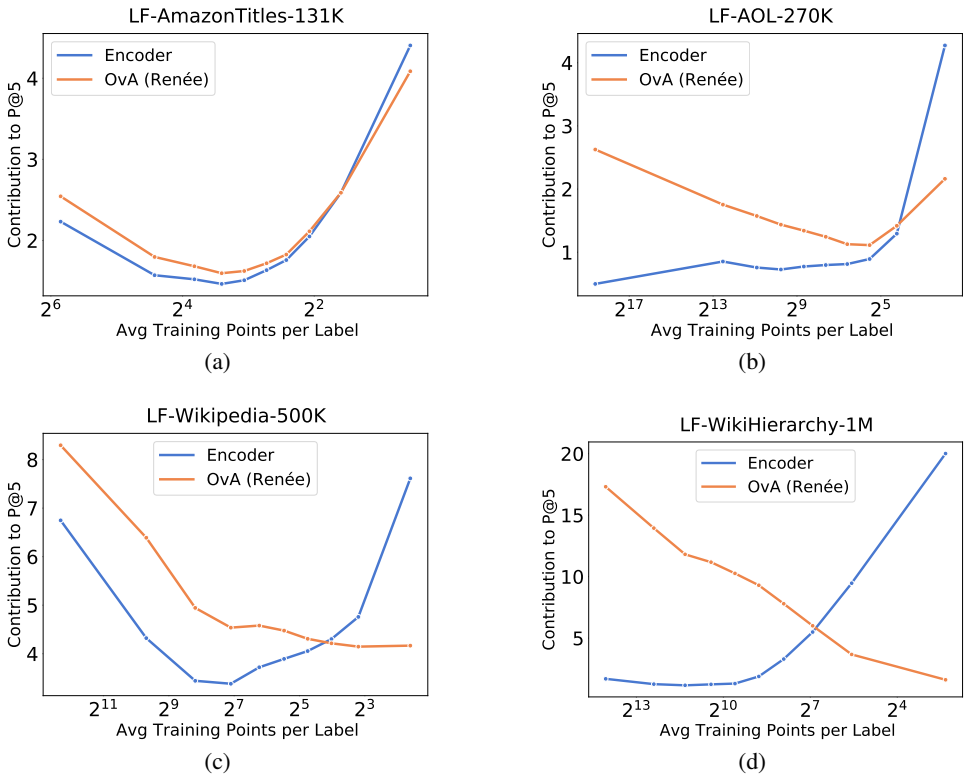

Figure 2: P@5 comparison of Siamese Encoder (blue) and OvA Classifier Renée (orange) on homogeneous (LF-AmazonTitles-131K: Fig. 2a, LF-Wikipedia-500K: Fig. 2c) and heterogeneous datasets (LF-AOL-270K: Fig. 2b, LF-WikiHierarchy-1M: Fig. 2d). Labels are partitioned into equi-volume bins based on their frequencies along the X-axis. The difference in performance (on both head and tail) is wider for heterogeneous datasets.

C.2 Comparison with SOTA and Tail XC methods

Table 5 demonstrates the enhanced performance achieved by applying LEVER to top-performing Extreme Classification (XC) methods, including ELIAS, CascadeXML, and Renée. On average PSP metrics are boosted by 5%, Coverage improves by 6.5% and Precision improves by 1.4%.

Table 6 presents a comparison of LEVER with various OvA-based methods (XR-Transformer, ELIAS, and CascadeXML) and Siamese encoder methods (NGAME and ECLARE). It's worth noting that for WikiHierarchy-1M, OvA and Siamese approaches exhibit significant trade-offs between precision and tail metrics.

C.3 Comparison with ensemble between tail Expert and OvA classifier

In the ensemble model, for each data point, the encoder and OvA model provide a shortlist of top-k labels along with their prediction scores. These two shortlists (containing a total of up to $2k$ labels) need to be combined into a single shortlist of k labels by tie-breaking as elaborated below. First, the labels with a frequency more than the cut-off are considered from the OvA's shortlist. Similarly, the labels with a frequency less than the cut-off are considered from the encoder's shortlist. Cut-offs are derived on the basis of the cross-over points between Encoder and Renée in the decile wise plots shown in Fig. 2. Then, the two resulting shortlists are combined by considering the assigned label scores from both models and retaining only the k overall highest-scoring labels. Table 7 shows that LEVER clearly outperforms the ensemble model in 3 out of 4 datasets across all metrics. In the case of LF-WikiHierarchy-1M, the ensemble model shows gains in coverage metrics ($\sim$4-5%), this comes at the expense of a significant loss in Precision ($\sim$30%). Figure 3 compares the performance

Table 5: Using LEVER with leading OvA approaches improves their tail label performance consistently across benchmarks, with an average gain of 5% in PSP and 6.5% in coverage (C), while maintaining comparable precision (P) with an average gain of 1.4%.

Model	LF-AmazonTitles-131K								
	P@1	**P@3**	**P@5**	**PSP@1**	**PSP@3**	**PSP@5**	**C@1**	**C@3**	**C@5**
ELIAS	37.28	25.18	18.14	28.95	34.45	39.08	23.73	42.36	49.06
ELIAS + LEVER	**42.86**	**28.37**	**20.16**	**36.30**	**41.05**	**45.43**	**29.81**	**49.88**	**56.25**
CascadeXML	35.96	24.77	18.15	26.22	33.06	38.72	21.14	40.41	48.38
CascadeXML + LEVER	**43.16**	**28.66**	**20.59**	**35.86**	**41.65**	**46.86**	**29.13**	**50.44**	**57.83**
Renée	46.05	30.81	**22.04**	38.47	44.87	**50.33**	31.31	53.50	61.03
Renée + LEVER	**46.44**	**30.83**	21.92	**39.70**	**45.44**	50.31	**32.82**	**55.11**	**61.94**
	LF-Amazon-131K								
ELIAS	43.03	29.27	21.20	33.49	40.80	46.76	27.04	49.10	57.34
ELIAS + LEVER	**47.38**	**32.24**	**23.22**	**38.97**	**46.74**	**52.79**	**31.47**	**55.27**	**63.40**
CascadeXML	43.76	29.75	21.63	34.05	41.69	48.09	30.94	54.20	62.74
CascadeXML + LEVER	**48.24**	**32.82**	**23.73**	**39.09**	**47.55**	**54.18**	**31.26**	**55.97**	**64.81**
Renée	48.05	32.33	23.26	39.32	47.10	53.51	31.49	55.81	64.61
Renée + LEVER	**49.19**	**33.30**	**24.04**	**40.64**	**48.48**	**54.87**	**32.39**	**56.81**	**65.20**
	LF-WikiSeeAlso-320K								
ELIAS	41.40	27.36	20.66	23.83	28.38	31.90	13.30	27.72	35.50
ELIAS + LEVER	**45.99**	**30.28**	**22.78**	**30.00**	**34.16**	**37.52**	**16.56**	**33.06**	**41.34**
CascadeXML	30.21	18.72	14.05	12.46	14.15	16.25	6.70	13.38	17.72
CascadeXML + LEVER	**38.84**	**25.43**	**19.36**	**21.62**	**25.85**	**29.45**	**12.01**	**25.12**	**32.59**
Renée	47.70	**31.90**	**23.82**	31.13	36.49	40.37	17.02	35.32	44.56
Renée + LEVER	**47.89**	31.52	23.53	**32.44**	**37.45**	**40.99**	**17.78**	**36.33**	**45.31**
	LF-Wikipedia-500K								
ELIAS	81.94	62.71	48.75	33.58	43.92	48.67	19.62	41.30	51.36
ELIAS + LEVER	**82.44**	**63.88**	**50.03**	**36.94**	**49.28**	**55.03**	**23.55**	**50.81**	**63.04**
CascadeXML	77.00	58.30	45.10	31.25	39.35	43.29	15.78	33.07	41.46
CascadeXML + LEVER	**80.10**	**60.41**	**46.44**	**36.79**	**46.65**	**50.99**	**23.99**	**49.16**	**60.13**
Renée	84.95	66.25	51.68	37.10	50.27	55.68	22.90	50.08	61.59
Renée + LEVER	**85.02**	**66.42**	**52.05**	**42.50**	**54.86**	**60.20**	**29.46**	**58.53**	**70.29**
	LF-AOL-270K								
ELIAS	40.83	22.33	14.91	13.29	21.46	25.22	10.46	22.85	27.06
ELIAS + LEVER	**40.85**	**22.83**	**15.57**	**13.68**	**24.30**	**30.43**	**10.52**	**26.33**	**33.56**
CascadeXML	**41.20**	**22.12**	14.82	**12.58**	19.53	23.19	**7.91**	22.09	29.83
CascadeXML + LEVER	39.41	21.78	**14.99**	11.96	**21.30**	**27.59**	7.86	**22.11**	**29.89**
Renée	40.97	23.34	15.85	15.06	26.36	31.97	12.40	29.77	36.53
Renée + LEVER	**41.70**	**24.76**	**17.07**	**20.38**	**37.07**	**45.13**	**17.43**	**42.54**	**52.01**
	LF-WikiHierarchy-1M								
ELIAS	**95.27**	**94.25**	**92.45**	17.15	24.41	30.01	4.00	7.78	10.49
ELIAS + LEVER	94.02	91.97	89.50	**28.27**	**36.80**	**42.13**	**10.78**	**18.88**	**23.03**
CascadeXML	**94.88**	**93.69**	**91.79**	16.03	22.87	28.17	3.12	6.17	8.52
CascadeXML + LEVER	94.77	93.54	91.56	**20.14**	**27.49**	**33.01**	**9.35**	**16.80**	**21.05**
Renée	95.01	**93.99**	**92.24**	19.69	27.36	33.20	6.62	11.39	14.56
Renée + LEVER	**95.19**	93.90	92.07	**24.79**	**32.74**	**38.29**	**9.08**	**16.12**	**20.02**
	LF-AmazonTitles-1.3M								
ELIAS	47.48	42.21	38.60	18.79	23.20	26.06	11.53	21.45	27.33
ELIAS + LEVER	**48.91**	**43.17**	**39.28**	**23.68**	**27.43**	**29.72**	**15.10**	**26.65**	**32.84**
CascadeXML	47.14	41.43	37.73	15.92	20.23	23.16	8.65	16.75	21.95
CascadeXML + LEVER	**47.98**	**42.02**	**38.12**	**20.06**	**24.51**	**27.28**	**12.36**	**22.57**	**28.52**
Renée	**56.10**	**49.91**	**45.32**	28.56	33.38	36.14	17.61	30.60	37.59
Renée + LEVER	56.01	49.43	44.85	**33.55**	**36.82**	**38.81**	**21.03**	**35.70**	**42.78**

of LEVER with the ensemble model and here we see a clear dip in the torso deciles. To better understand why the ensemble curve doesn't exactly mimic the OvA curve before the cutoff and encoder curve after the cutoff, consider the following toy example:

Assume a dataset D with 8 labels which are partitioned into 3 deciles (head, torso, and tail deciles). Out of 8 labels, 3 belong to the head decile (H_1, H_2, H_3), 2 belong to the torso decile (O_1, O_2) and the remaining 3 belong to the tail decile (T_1, T_2, T_3). The cut-off threshold partitions the label set into 2 sets: (i) labels with frequency greater than cut-off: (H_1, H_2, H_3, O_2) and labels

Table 6: Comparison between LEVER and leading OvA-based methods such as XR-Transformer, ELIAS, and CascadeXML, as well as Siamese encoder-based methods like NGAME and ECLARE. Note that for LF-WikiHierarchy-1M, OvA and Siamese-based methods display significant trade-offs between precision and tail metrics. Siamese-based methods score much higher in PSP numbers (+16 on average) but lag behind in precision (-14 on average) when compared to OvA-based methods.

		P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	C@1	C@3	C@5
		LF-AmazonTitles-131K								
SOTA XC Methods	XR-Transformer	38.10	25.57	18.32	28.86	34.85	39.59	20.24	40.70	48.87
	ELIAS	37.28	25.18	18.14	28.95	34.45	39.08	23.73	42.36	49.06
	CascadeXML	35.96	24.77	18.15	26.22	33.06	38.72	21.14	40.41	48.38
	ECLARE	40.74	27.54	19.88	33.51	39.55	44.70	20.57	44.45	53.89
	NGAME	46.01	30.28	21.47	38.81	44.40	49.43	32.04	53.61	60.65
	Renée	46.05	30.81	**22.04**	38.47	44.87	50.33	31.31	53.50	61.03
Tail XC Methods	Renée +TAUG	44.34	29.73	21.15	36.49	42.83	47.85	29.47	51.52	58.68
	Renée + BoW	42.95	29.18	21.03	36.96	42.86	48.09	30.03	51.78	59.17
	Renée + L2Reg	45.19	29.92	21.29	38.74	44.53	49.49	31.66	53.65	60.80
	Renée + GLaS	45.35	30.03	21.33	38.74	44.53	49.49	31.90	54.02	61.15
	Renée + Gandalf	45.86	30.53	21.79	**40.49**	**45.83**	**50.96**	**33.17**	**55.36**	**62.22**
	Renée + LEVER	**46.44**	**30.83**	21.92	39.70	45.44	50.31	32.82	55.11	61.94
		LF-AOL-270K								
SOTA XC Methods	XR-Transformer	37.56	20.44	13.94	11.76	21.10	26.31	8.83	23.07	29.44
	ELIAS	40.83	22.33	14.91	13.29	21.46	25.22	10.46	22.85	27.06
	CascadeXML	41.20	22.12	14.82	12.58	19.53	23.19	7.91	22.09	29.83
	ECLARE	28.53	16.18	11.55	10.11	18.69	24.58	7.41	20.59	28.11
	NGAME	9.44	22.93	15.30	16.37	29.12	36.29	14.28	34.70	43.84
	Renée	40.97	23.34	15.85	15.06	26.36	31.97	12.40	29.77	36.53
Tail XC Methods	Renée +TAUG	40.40	22.80	15.56	15.72	26.74	32.35	12.46	29.26	35.88
	Renée + BoW	41.11	23.91	16.41	15.58	30.28	37.90	12.67	34.32	43.45
	Renée + L2Reg	39.83	21.75	14.71	12.21	20.09	24.36	8.67	21.07	26.27
	Renée + GLaS	40.91	23.25	15.78	14.67	26.11	31.75	12.36	29.41	36.06
	Renée + Gandalf	40.63	23.01	15.58	15.10	26.64	32.17	12.63	29.82	36.31
	Renée + LEVER	**41.71**	**24.77**	**17.07**	**20.38**	**37.07**	**45.14**	**17.43**	**42.54**	**52.01**
		LF-Wikipedia-500K								
SOTA XC Methods	XR-Transformer	81.62	61.38	47.85	33.58	42.97	47.81	19.05	40.05	50.66
	ELIAS	81.94	62.71	48.75	33.58	43.92	48.67	19.62	41.30	51.36
	CascadeXML	77.00	58.3	45.10	31.25	39.35	43.29	15.78	33.07	41.46
	NGAME	84.01	64.69	49.97	41.25	52.57	57.04	26.22	51.42	64.79
	Renée	84.95	66.25	51.68	41.25	52.57	57.04	22.90	50.08	61.59
Tail XC Methods	Renée +TAUG	83.07	64.46	50.32	33.76	46.54	52.16	19.88	44.74	56.13
	Renée + BoW	84.43	66.09	51.74	36.66	49.79	55.55	22.92	49.64	61.40
	Renée + L2Reg	84.57	66.05	51.50	39.55	52.42	57.43	26.52	53.95	65.14
	Renée + GLaS	84.85	**66.63**	**52.09**	37.27	51.54	57.15	23.43	52.02	63.90
	Renée + Gandalf	84.59	66.07	51.63	37.05	49.94	55.31	23.09	49.87	61.24
	Renée + LEVER	**85.02**	66.42	52.05	**42.50**	**54.86**	**60.20**	**29.46**	**58.53**	**70.29**
		LF-WikiHierarchy-1M								
SOTA XC Methods	XR-Transformer	95.33	**94.48**	**92.66**	15.96	21.13	28.76	2.98	6.23	8.86
	ELIAS	95.27	94.25	92.45	17.15	24.41	30.01	4.00	7.78	10.49
	CascadeXML	94.88	93.69	91.79	16.03	22.87	28.17	3.12	6.17	8.52
	ECLARE	91.24	89.60	87.39	15.46	22.31	27.24	2.49	5.82	9.17
	NGAME	83.16	78.24	73.90	**38.43**	**44.22**	**47.93**	7.83	**22.59**	**29.25**
	Renée	95.01	93.99	92.24	19.69	27.36	33.20	6.62	11.39	14.56
Tail XC Methods	Renée +TAUG	**95.34**	94.45	92.27	16.95	24.06	29.69	3.59	7.19	9.94
	Renée + BoW	93.92	92.04	90.27	24.25	31.10	36.30	7.84	14.77	18.39
	Renée + L2Reg	94.36	93.22	91.34	18.44	25.79	31.37	5.55	9.89	12.91
	Renée + GLaS	95.01	93.98	92.26	20.07	27.82	33.70	6.89	11.82	15.08
	Renée + Gandalf	93.01	90.85	88.16	21.84	30.05	36.09	6.92	13.17	17.52
	Renée + LEVER	95.19	93.90	92.07	24.79	32.74	38.29	**9.08**	16.12	20.02

with frequency less than cut-off: (O_1, T_1, T_2, T_3). Assume a data point d has ground truth labels: $(H_1, H_2, O_1, O_2, T_1)$.

Below we list the predictions of different models in the format of "label ID:model score"

Top-5 encoder predictions $(T_1 : 0.8, T_2 : 0.6, T_3 : 0.4, O_1 : 0.2, O_2 : 0.1)$.

Table 7: Comparison of LEVER with an ensemble of OvA and tail expert encoder. LEVER outperforms the ensemble consistently on all metrics for 3 out of 4 datasets . Note that for LF-WikiHierarchy-1M even though the ensemble improves coverage, the drop in precision is very large (31% on average).

	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	C@1	C@3	C@5
					LF-AmazonTitles-131K				
Ensemble	42.98	26.84	18.23	37.24	41.58	44.69	30.38	51.57	57.54
Renée + LEVER	**46.44**	**30.83**	**21.92**	**39.70**	**45.44**	**50.31**	**32.82**	**55.11**	**61.94**
					LF-AOL-270K				
Ensemble	35.20	20.33	13.69	19.43	36.98	44.74	17.19	44.80	54.87
Renée + LEVER	**41.71**	**24.77**	**17.07**	**20.38**	**37.07**	**45.14**	**17.43**	**42.54**	**52.01**
					LF-Wikipedia-500K				
Ensemble	82.55	61.96	46.65	39.82	51.29	55.76	25.72	58.46	**72.17**
Renée + LEVER	**85.02**	**66.42**	**52.05**	**42.50**	**54.86**	**60.20**	**29.46**	**58.53**	70.29
					LF-WikiHierarchy-1M				
Ensemble	67.48	62.65	58.39	**28.08**	31.75	34.01	**10.86**	**20.82**	**25.89**
Renée + LEVER	**95.19**	**93.90**	**92.07**	24.79	**32.74**	**38.29**	9.08	16.12	20.02

Top-5 OvA predictions ($H_1 : 0.7, H_2 : 0.5, H_3 : 0.3, O_2 : 0.2, O_1 : 0.1$)

To compute the ensemble model predictions, we first restrict the predictions of the individual models based on the cutoff frequency, i.e Encoder's predictions are restricted to (O_1, T_1, T_2, T_3) and OvA predictions are restricted to (H_1, H_2, H_3, O_2). This gives the following filtered shortlists:

Encoder: $(T_1 : 0.8, T_2 : 0.6, T_3 : 0.4, O_1 : 0.2)$

OvA: $(H_1 : 0.7, H_2 : 0.5, H_3 : 0.3, O_2 : 0.2)$

Next, we combine and sort the labels based on the scores from both the encoder and OvA as follows:

$(T_1 : 0.8, H_1 : 0.7, T_2 : 0.6, H_2 : 0.5, T_3 : 0.4, H_3 : 0.3, O_2 : 0.2, O_1 : 0.2)$

Finally, we retain only the top-5 highest scoring labels as our final ensemble predictions:

Ensemble predictions: $(T_1 : 0.8, H_1 : 0.7, T_2 : 0.6, H_2 : 0.5, T_3 : 0.4)$ Table 8 shows the contribution to P@5 for different models across the three deciles. Note that the example is in line with our observations in Fig. 3 where (i) Encoder performs better on tail deciles (blue curve), (ii) OvA models perform better on head deciles (orange), (iii) Ensemble (green) between Encoder and OvA models perform comparably to Encoders on tail deciles and OvA based models on head deciles but incurs significant losses in torso deciles, (iv) Performance of the ensemble model can be worse than the individual models (e.g. ensemble P@5 $<$ OvA P@5 in toy example). If the Ensemble model were to dominate both Encoder and OvA models it should have achieved decile-wise contributions of $(2/5, 2/5, 1/5)$ which is not the case. On average, the torso labels are ranked relatively lower by both models since neither model specializes in them. Further, when combined using the proposed ensemble these labels get more aggressively down-voted.

Table 8: P@5 performance for different models across deciles.

Model	Head Decile P@5	Torso Decile P@5	Tail Decile P@5	Overall P@5
Encoder	0/5	2/5	1/5	3/5
OvA	2/5	2/5	0/5	4/5
Ensemble	2/5	0/5	1/5	3/5

C.4 COMPARISON WITH NGAME ENCODER

C.5 EFFECT OF RE-RANKING ON LEVER AND OTHER TAIL XC APPROACHES

Table 10 illustrates the impact of post-hoc reranking using inverse propensity scores, on LEVER and other Tail XC approaches. The application of reranking shows varying degrees of trade-offs between precision and tail metrics across different models. Notably, while LEVER attains superior

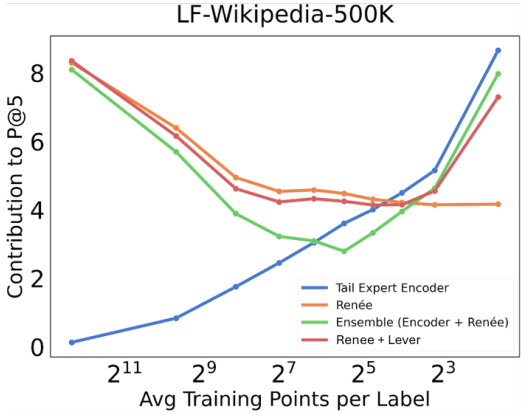

Figure 3: Performance comparison of a Tail Expert Encoder (blue), an OvA Classifier (orange), an Ensemble of Expert Encoder and OvA Classifier (Green) and LEVER-based OvA Classifier (red) in the presence of label skew. Labels are partitioned into equi-volume bins based on their frequencies along the X-axis. OvA overfits to tail labels with few training points. Encoder leverages label metadata to improve on tail but underfits to head. Ensemble mode suffers on torso labels. LEVER combines the strengths of both OvA and Encoder to perform well on all labels.

Table 9: Renée (OvA) and Siamese trained NGAME encoder exhibit different trade-offs on in precision and tail-metrics (PSP, Coverage). LEVER improves the tail performance of Renée (+5% on average in PSP and +3% on average in Coverage) while retaining comparable precision.

	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	C@1	C@3	C@5
					LF-AmazonTitles-131K				
NGAME Encoder	41.33	28.71	20.77	39.24	44.62	49.52	32.83	55.11	61.95
Renée	46.05	30.81	22.04	38.47	44.87	50.33	31.31	53.50	61.03
Renée + LEVER	46.44	30.83	21.92	39.70	45.44	50.31	32.82	55.11	61.94
					LF-AOL-270K				
NGAME Encoder	23.24	15.67	11.68	25.41	36.24	43.43	27.49	47.45	55.58
Renée	40.97	23.34	15.85	15.06	26.36	31.97	12.40	29.77	36.53
Renée + LEVER	41.71	24.77	17.07	20.38	37.07	45.14	17.43	42.54	52.01
					LF-Wikipedia-500K				
NGAME Encoder	67.81	45.65	34.31	60.76	57.25	57.20	48.76	71.41	78.59
Renée	84.95	66.25	51.68	37.10	50.27	55.68	22.90	50.08	61.59
Renée + LEVER	85.02	66.42	52.05	42.50	54.86	60.20	29.46	58.53	70.29
					LF-WikiHierarchy-1M				
NGAME Encoder	66.82	60.64	55.42	75.63	73.02	70.54	21.82	42.93	50.62
Renée	95.01	93.99	92.24	19.69	27.36	33.20	6.62	11.39	14.56
Renée + LEVER	95.19	93.90	92.07	24.79	32.74	38.29	9.08	16.12	20.02

performance in tail metrics for three out of four datasets, there is a trade-off in precision compared to other methods in the LF-WikiHierarchy-1M dataset.

C.6 Ablations

Effect of Teacher Model: Table 11 demonstrates the impact of employing various encoders as a tail expert. We conduct a comparison with two alternative encoders: (i) MiniLM, a 3-layer transformer model, and (ii) Astec, which learns a projection matrix from sparse Bag of Words (BoW) features to a dense embedding space. The results highlight that the choice of a superior teacher substantially enhances the performance of LEVER.

Effect of sampling strategy: LEVER makes use of NGAME Module (Dahiya et al., 2022) trained using mini-batches of labels instead of documents. The modification helps specialize the Siamese encoder towards tail labels. Renée + LEVER$_{doc}$ denotes the model that uses NGAME encoder with mini-batches of documents to augment the training data. Table 12 shows the effect of sampling

Table 10: Performance comparison of LEVER with other tail XC approaches LEVER outperforms other tail XC methods in tail metrics on 3 out of 4 datasets. while LEVER attains superior performance in tail metrics for three out of four datasets, there is a trade-off in precision compared to other methods in the LF-WikiHierarchy-1M dataset.

Dataset	Model	P@1	P@3	P@5	Ps@1	Ps@3	Ps@5	C@1	C@3	C@5
LF-AmazonTitles-131K	Renée	46.05	30.81	22.04	38.47	44.87	50.33	31.31	53.50	61.03
	+ Rerank	**46.16**	**30.80**	**22.02**	39.99	45.53	50.78	32.90	54.65	61.84
	+ L2Reg + ReRank	44.89	29.71	21.14	39.99	44.58	49.29	33.18	54.48	61.19
	+ GLaS + ReRank	45.06	29.82	21.17	40.18	44.83	49.49	33.36	54.78	61.48
	+ Gandalf + ReRank	44.17	30.29	21.90	40.98	**46.09**	**51.19**	33.61	**56.27**	**62.97**
	+ LEVER + ReRank	45.36	30.67	21.95	**41.13**	46.00	50.85	**33.91**	55.79	62.31
LF-AOL-270K	Renée	40.97	23.34	15.85	15.06	26.36	31.97	12.40	29.77	36.53
	+ Rerank	**41.53**	24.11	16.44	20.21	31.11	37.24	20.27	36.13	43.01
	+ L2Reg + ReRank	40.25	22.43	15.25	15.16	23.59	28.81	13.69	26.38	32.56
	+ GLaS + ReRank	41.41	24.01	16.37	19.93	30.71	36.82	20.05	35.71	42.56
	+ Gandalf + ReRank	40.87	23.49	15.94	20.70	30.68	36.22	20.61	35.47	41.65
	+ LEVER + ReRank	39.60	**24.23**	**16.92**	**28.20**	**41.58**	**49.33**	**27.40**	**48.95**	**57.62**
LF-Wikipedia-500K	Renée	84.95	66.25	51.68	37.10	50.27	55.68	22.90	50.08	61.59
	+ Rerank	79.28	63.56	50.80	53.44	56.16	59.06	41.58	59.52	67.17
	+ L2Reg + ReRank	79.30	63.64	50.69	57.67	58.06	60.32	45.03	62.90	70.14
	+ GLaS + ReRank	80.20	**64.74**	**51.50**	53.22	56.85	60.07	41.49	60.17	68.55
	+ Gandalf + ReRank	**80.58**	64.55	51.29	51.03	55.09	58.36	39.07	58.00	66.23
	+ LEVER + ReRank	75.34	62.07	50.26	**59.15**	**60.29**	**62.95**	**47.24**	**68.40**	**75.94**
LF-WikiHierarchy-1M	Renée	95.01	93.99	92.24	19.69	27.36	33.20	6.62	11.39	14.56
	+ Rerank	89.95	89.94	88.86	44.15	52.89	58.47	18.15	30.53	35.16
	+ L2Reg + ReRank	91.18	90.92	89.51	41.22	49.41	55.00	16.34	27.78	32.73
	+ GLaS + ReRank	**93.09**	**92.42**	**91.00**	46.38	54.75	60.17	18.78	31.41	36.07
	+ Gandalf + ReRank	86.03	83.04	80.88	51.77	57.79	61.37	**20.21**	34.82	39.73
	+ LEVER + ReRank	86.27	84.51	83.69	**52.17**	**58.92**	**63.59**	19.60	**34.84**	**40.58**

Table 11: Comparison of different encoders: a 3-layer MiniLM and Astec, and their effects on LEVER performance. The Astec encoder learns a projection matrix that maps sparse Bag-of-Words features to a dense embedding space. The table below shows that a superior expert encoder leads to improved performance in both Precision and tail metrics, namely PSP and coverage.

	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	C@1	C@3	C@5
	LF-AmazonTitles-131K								
Astec Encoder	19.78	18.28	14.39	16.96	27.38	33.61	14.34	35.77	44.37
MinLM-L3 Encoder	23.86	21.65	16.82	20.22	32.44	39.26	17.20	41.80	50.82
NGAME Encoder	41.33	28.71	20.77	39.24	44.62	49.52	32.83	55.11	61.95
Renée	46.05	30.81	22.04	38.47	44.87	50.33	31.31	53.50	61.03
Renée + LEVER (Astec)	42.76	28.97	20.97	36.09	42.25	47.82	29.54	51.29	59.01
Renée + LEVER (MiniLM-L3)	45.26	30.40	21.82	38.28	44.67	50.09	31.22	53.73	61.11
Renée + LEVER (NGAME)	46.44	30.83	21.92	39.70	45.44	50.31	32.82	55.11	61.94
	LF-AmazonTitles-1.3M								
MinLM-L3 Encoder	36.14	30.25	26.32	28.12	29.00	29.29	18.38	31.56	37.83
Astec Encoder	32.10	26.86	23.43	25.48	26.20	26.48	16.81	29.32	35.46
NGAME Encoder	42.27	36.16	31.63	35.62	38.11	38.87	22.37	38.93	46.98
Renée	56.10	49.91	45.32	28.56	33.38	36.14	17.61	30.60	37.59
Renée + LEVER (Astec)	49.30	43.12	39.26	30.46	33.83	35.86	18.39	33.10	40.71
Renée + LEVER (MiniLM-L3)	50.24	44.01	40.08	32.73	35.90	37.73	20.09	35.55	43.23
Renée + LEVER (NGAME)	56.01	49.43	44.85	33.55	36.82	38.81	21.03	35.70	42.78

strategy by comparing Renée + LEVER$_{doc}$ and Renée + LEVER. Renée + LEVER outperforms Renée + LEVER$_{doc}$ by upto 2% in PSP while being comparable in precision.

Effect of varying τ: The hyperparameter τ is tuned using a validation set that contains 5% of the training data. The best value of τ obtained is then used to train LEVER on complete training data. Fig. 4 shows the effect of varying τ on LEVER's performance. It can be seen that increasing τ leads to better performance on tail labels, while it hurts the head or torso labels.

Table 12: Comparison of Renée + LEVER$_{doc}$ and Renée + LEVER on Different Datasets

Dataset	Model	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
LF-AmazonTitles-131K	Renée + LEVER$_{doc}$	46.05	30.81	**22.04**	38.47	44.87	**50.33**
	Renée + LEVER	**46.44**	**30.83**	21.92	**39.70**	**45.44**	50.31
LF-Wikipedia-500K	Renée + LEVER$_{doc}$	**85.09**	65.94	51.69	40.17	52.65	58.18
	Renée + LEVER	85.02	**66.42**	**52.05**	**42.50**	**54.86**	**60.20**
LF-WikiHierarchy-1M	Renée + LEVER$_{doc}$	95.02	**94.06**	**92.28**	23.64	31.28	36.89
	Renée + LEVER	**95.19**	93.90	92.07	**24.79**	**32.74**	**38.29**
LF-AmazonTitles-1.3M	Renée + LEVER$_{doc}$	55.02	48.94	44.82	31.86	36.42	38.75
	Renée + LEVER	**56.01**	**49.43**	**44.85**	**33.55**	**36.82**	**38.81**

Table 13: P and PSP Comparison of NGAME, Renée, and Renée + LEVER on QK-20M Dataset

	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
NGAME	69.94	52.72	44.81	48.24	55.63	58.71
Renée	**72.14**	**54.87**	**47.02**	50.75	58.90	62.56
Renée + LEVER	71.70	54.48	46.56	**54.74**	**63.36**	**67.14**

D Model Details and Hyperparameters

D.1 Tail expert Siamese Encoder

NGAME's (Dahiya et al., 2022) hyperparameters include:

- `cluster-sz`: Mini-batches in NGAME are created from clusters of similar documents (or labels). To build a batch of B documents (or labels) we pick $B/$`cluster-sz` clusters.
- `cluster-freq`: Denotes the frequency of refreshing the clusters using updated embeddings.
- γ: Denotes the margin enforced while training with contrastive loss.
- `lr`: Learning rate for the encoder.
- `bsz`: Denotes the size of mini-batches.
- `epochs`: Denotes the number of epochs for which the NGAME module is trained.

To train the tail-expert NGAME module we closely follow the settings from (Dahiya et al., 2022). NGAME utilizes a 6-layer DistilBERT architecture. Table 14 shows the hyperparameters used on benchmark as well as newly contributed datasets.

Table 14: Hyperparameters of tail-expert NGAME module. × indicates use of random mini-batches.

Dataset	`cluster-sz`	`cluster-freq`	γ	LR	`bsz`	`Epochs`
LF-AmazonTitles-131K	8	5	0.3	2×10^{-4}	1600	300
LF-Amazon-131K	512	5	0.3	2×10^{-4}	700	400
LF-AOL-270K	×	×	0.05	2×10^{-4}	3200	300
LF-WikiSeeAlso-320K	512	5	0.3	2×10^{-4}	1024	300
LF-Wikipedia-500K	16	5	0.3	2×10^{-4}	512	40
LF-WikiHierarchy-1M	1024	5	0.3	2×10^{-4}	6400	300
LF-AmazonTitles-1.3M	8	5	0.3	2×10^{-4}	1600	400

D.2 ELIAS

ELIAS's (Gupta et al., 2022) hyperparameters include:

- C: Denotes the number of clusters in the index graph.

- α: Multiplicative hyperparameter that controls the effective number of clusters that can get activated for a given input get activated for a given input.
- β: Multiplicative hyperparameter that controls the effective number of labels that can get assigned to a particular cluster.
- ρ: Controls the row-wise sparsity of the adjacency matrix.
- λ: Controls importance of classification loss $\mathcal{L}_c$ and shortlist loss $\mathcal{L}_s$ in the final loss.
- K: Denotes the shortlist size, label classifiers are only evaluated on top-K shortlisted labels.
- b: Denotes the beam size.
- `epochs`: Denotes the total number of epochs (i.e. including stage 1 and stage 2 training).
- LR_ϕ, LR_W: Denotes the learning rate used for the transformer encoder and the rest of the model.
- `bsz`: denotes the batch-size of the mini-batches used during training

We closely follow the setting used in (Gupta et al., 2022). ELIAS uses a 6-layer Distil-BERT encoder. Note that the NGAME encoder is only used to augment the ground truth with labels similar to a particular label, it is not used in any other way while training ELIAS. Table 15 shows the hyperparameters used on the benchmark as well as newly contributed datasets.

Table 15: Hyperparameters of ELIAS

Dataset	C	α	β	ρ	λ	K	b	Epochs	LR_ϕ	LR_W	bsz
LF-AmazonTitles-131K	2048	10	150	1000	0.05	2000	20	60	1×10^{-4}	2×10^{-2}	512
LF-Amazon-131K	2048	10	150	1000	0.05	2000	20	70	7×10^{-5}	5×10^{-3}	1024
LF-AOL-270K	4096	10	150	1000	0.05	2000	20	70	3×10^{-5}	1×10^{-3}	8192
LF-WikiSeeAlso-320K	4096	10	150	1000	0.05	2000	20	40	5×10^{-5}	5×10^{-3}	1024
LF-Wikipedia-500K	8192	10	150	1000	0.05	2000	20	40	5×10^{-5}	5×10^{-3}	256
LF-WikiHierarchy-1M	16384	10	150	1000	0.05	2000	20	30	5×10^{-5}	5×10^{-3}	1024
LF-AmazonTitles-1.3M	16384	10	150	1000	0.05	2000	20	40	2×10^{-5}	1×10^{-3}	1024

D.3 CASCADEXML

CascadeXML's (Kharbanda et al., 2022) hyperparameters include:

- `Ep`: Number of epochs CascadeXML is trained for.
- `bsz`: Denotes the batch size used for training.
- `label resolution`: Denotes the BERT layers and clustering size used at each resolution.
- `dropout`: Dropout used at each resolution.
- `shortlist size`: Cluster size used at each resolution.
- LR_ϕ, LR_W: Denotes the learning rate used for the transformer encoder and weight vectors.

Table 16: Hyperparameters of CascadeXML

Dataset	Ep	bsz	Label Resolution	Dropout	Shortlist-sz	LR_ϕ	LR_W
LF-AmazonTitles-131K	15	64	{5,6}:2^{10} — {8}:2^{13} — {10}:2^{16} — 12 : 131073	0.2, 0.25, 0.35, 0.5	$2^{10}, 2^{10}, 2^{10}$	$1e^{-4}$	$1e^{-3}$
LF-Amazon-131K	15	64	{5,6}:2^{9} — {8}:2^{12} — {10}:2^{15} — 12 : 131073	0.2, 0.25, 0.4, 0.5	$2^{6}, 2^{7}, 2^{8}$	$1e^{-4}$	$1e^{-3}$
LF-AOL-270K	12	96	{5,6}:2^{10} — {8}:2^{13} — {10}:2^{16} — 12 : 272825	0.2, 0.25, 0.35, 0.5	$2^{10}, 2^{10}, 2^{10}$	$1e^{-4}$	$1e^{-3}$
LF-WikiSeeAlso-320K	12	64	{5,6}:2^{10} — {8}:2^{13} — {10}:2^{16} — 12 : 312330	0.2, 0.25, 0.35, 0.5	$2^{10}, 2^{11}, 2^{12}$	$1e^{-4}$	$1e^{-3}$
LF-Wikipedia-500K	12	256	{5,6}:2^{10} — {8}:2^{13} — {10}:2^{16} — 12 : 501070	0.2, 0.25, 0.35, 0.5	$2^{10}, 2^{10}, 2^{11}$	$1e^{-4}$	$1e^{-3}$
LF-WikiHierarchy-1M	12	96	{5,6}:2^{10} — {8}:2^{13} — {10}:2^{16} — 12 : 976214	0.2, 0.25, 0.35, 0.5	$2^{10}, 2^{10}, 2^{10}$	$1e^{-4}$	$1e^{-3}$
LF-AmazonTitles-1.3M	10	48	{7,8}:2^{13} — {10}:2^{16} — 12 : 1305265	0.2, 0.3, 0.4	$2^{10}, 2^{11}$	$1e^{-4}$	$1e^{-3}$

We closely follow the setting used in (Kharbanda et al., 2022). CascadeXML uses a 12-layer BERT encoder. Note that the NGAME encoder is only used to augment the ground truth with labels similar to a particular label, it is not used in any other way while training CascaseXML. Table 16 shows the hyperparameters used on benchmark as well as newly contributed datasets.

D.4 RENÉE

Renée's (Jain et al., 2023) hyperparameters include:

- `epochs`: Denotes the total number of epochs for which Renée is trained.
- `dropout`: Denotes the probability of randomly dropping the encoder outputs in order to regularise the network.
- `warmup`: Warmup steps is the number of training iterations over which both the encoder and the classifier learning rates are linearly increased from 0 to the maximum value.
- LR_ϕ, LR_W: Denotes the learning rate used for the transformer encoder and the classifier layer.
- `bsz`: Denotes the batch size of the mini-batches used during training.
- `clf-wd`: Weight decay for fully connected layer parameters.

Table 17: Hyperparameters of Renée

Dataset	Epochs	Dropout	Warmup	LR_ϕ	LR_W	bsz	clf-wd
LF-AmazonTitles-131K	100	0.85	5000	1×10^{-5}	5×10^{-2}	512	1×10^{-4}
LF-Amazon-131K	100	0.85	5000	1×10^{-5}	5×10^{-2}	512	1×10^{-4}
LF-AOL-270K	100	0.60	20000	1×10^{-6}	1×10^{-3}	1024	1×10^{-4}
LF-WikiSeeAlso-320K	100	0.75	5000	2×10^{-4}	2×10^{-1}	2048	1×10^{-4}
LF-Wikipedia-500K	100	0.70	5000	5×10^{-5}	4×10^{-3}	2048	1×10^{-4}
LF-WikiHierarchy-1M	100	0.70	20000	1×10^{-4}	2×10^{-3}	1024	1×10^{-2}
LF-AmazonTitles-1.3M	100	0.70	15000	1×10^{-6}	1×10^{-2}	1024	1×10^{-4}

We closely follow the setting used in (Jain et al., 2023). Renée uses a 6-layer Distil-BERT encoder. Table 17 shows the hyperparameters used on benchmark as well as newly contributed datasets.

D.5 RERANK + TAUG

ReRank + TAUG (Wei et al., 2021) hyperparameters include:

- ϵ_{split}: Denotes the proportion of labels that will be considered as head labels. The original dataset D containing L labels is split into 2 datasets D_h and D_t. D_h contains headmost $\epsilon_{split}L$ labels and their associated training points, while D_t contains the remaining $L - \epsilon_{split}L$ labels along with their associated training points.
- `n-aug`: Denotes the number of additional data points that will be generated for each data point in D_t.
- p_{drop}: Denotes the probability of dropping a token from the data point.
- p_{swap}: Denotes the probability of swapping two randomly chosen tokens.
- `rerank-strategy`: Denotes the multiplicative factor used to re-rank scores. We use the label inverse propensity factor to perform re-ranking.

Table 18: Hyperparameters of Re-rank + TAUG

Dataset	ϵ_{split}	n-aug	p_{drop}	p_{swap}
LF-AmazonTitles-131K	0.90	8	0.30	0.30
LF-AOL-270K	0.65	6	0.20	0.20
LF-Wikipedia-500K	0.90	4	0.10	0.10
LF-WikiHierarchy-1M	0.90	4	0.20	0.20

D.6 GANDALF

Gandalf's (Kharbanda et al., 2023) hyperparameters include:

- `threshold`: Denotes the threshold used to filter out labels obtained from the normalized label correlation graph during augmentation.

We closely follow the settings used in (Kharbanda et al., 2023) and use threshold of 0.1 for all datasets.

D.7 LEVER

LEVER uses the parameter τ to control the number of entities (data points or labels) are added for each label. We only add entities having cosine similarity greater than 0.8 with the target label. The hyper-parameters m, c of Theorem 2 were set to 1,0 as these worked consistently well across datasets. Table 20 shows the value of τ for benchmark and newly contributed datasets. Table 19 compares the training time of the base OvA classifier with its LEVER regularized counterpart. LEVER increases the train time to 1.7x times the original OvA classifier in the worst case.

Table 19: Training time (in hours) for different models on a single NVIDIA V100 GPU. In the worst case, LEVER increases the train time to 1.7x times the original classifier.

Dataset	**ELIAS**	**ELIAS + LEVER**	**CascadeXML**	**CascadeXML + LEVER**	**Renée**	**Renée + LEVER**
LF-AmazonTitles-131K	4.33	6.95	3.62	5.25	5.77	8.22
LF-Amazon-131K	19.44	31.11	4.60	6.80	8.33	12.00
LF-AOL-270K	60.66	63.00	42.12	43.80	28.20	30.01
LF-WikiSeeAlso-320K	25.33	41.33	12.40	18.40	16.56	25.07
LF-Wikipedia-500K	138.60	176.10	29.58	39.00	104.66	133.33
LF-WikiHierarchy-1M	24.00	40.80	9.85	15.54	12.11	19.77
LF-AmazonTitles-1.3M	40.00	64.00	70.00	108.70	60.88	92.22

Table 20: LEVER's Hyperparameter τ on Different Datasets

Dataset	τ
LF-AmazonTitles-131K	15
LF-Amazon-131K	20
LF-AOL-270K	100
LF-WikiSeeAlso-320K	4
LF-Wikipedia-500K	45
LF-WikiHierarchy-1M	4
LF-AmazonTitles-1.3M	15

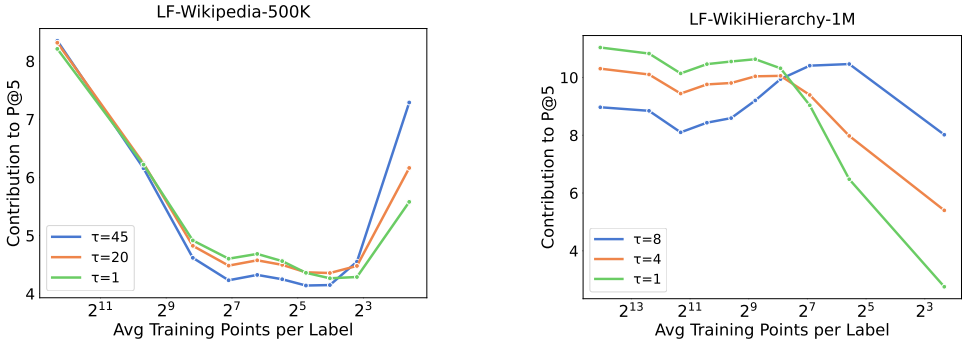

Figure 4: Effect of varying hyperparameter τ on LEVER's head and tail performance on LF-Wikipedia-500K and LF-WikiHierarchy-1M.