EFFICACY OF DUAL-ENCODERS FOR EXTREME MULTI-LABEL CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Dual-encoder models have demonstrated significant success in dense retrieval tasks for open-domain question answering that mostly involves zero-shot and few-shot scenarios. However, their performance in many-shot retrieval problems where training data is abundant, such as extreme multi-label classification (XMC), remains under-explored. Existing empirical evidence suggests that, for such problems, the dual-encoder method's accuracies lag behind the performance of state-of-the-art (SOTA) extreme classification methods that grow the number of learnable parameters linearly with the number of classes. As a result, some recent extreme classification techniques like (Dahiya et al., 2023a;b) use a combination of dual-encoders and a learnable classification head for each class to excel on these tasks. In this paper, we investigate the potential of "pure" DE models in XMC tasks. Our findings reveal that when trained correctly standard dual-encoders can match or outperform SOTA extreme classification methods by up to 2% at Precision@1 even on the largest XMC datasets while being $20\times$ smaller in terms of the number of trainable parameters. We further propose a differentiable top-k error-based loss function, which can be used to specifically optimize for Recall@k metrics. We include our `PyTorch` implementation along with other resources for reproducing the results in the supplementary material.

1 INTRODUCTION

Dual-encoder (DE) models have been highly successful in dense retrieval tasks for open-domain question answering (openQA) systems (Lee et al., 2019; Karpukhin et al., 2020; Qu et al., 2021), where they efficiently retrieve relevant passages or documents from a large corpus given a user's query. These models excel at leveraging their learned representations to identify and rank pertinent documents, by mapping both queries and documents into a shared embedding space, enabling efficient and effective retrieval using *fast similarity search methods* (Guo et al., 2020; Johnson et al., 2021). So far, DE models have been primarily explored in the zero-shot and the few-shot scenarios, where the models are required to *generalize* and retrieve relevant documents even though they might not have appeared in the training set.

An important retrieval scenario that frequently arises in real-world applications such as search engines (Mitra & Craswell, 2018) and recommendation systems (Covington et al., 2016; Zhang et al., 2019) is where we want to perform retrieval for documents/items from a large collection based on a significant number of seen examples per document/item, i.e., a *many-shot scenario*. Such tasks are often formulated as extreme multi-label classification (XMC) task, where each document/item to be retrieved is considered as a separate label (Bhatia et al., 2016). Typically, XMC algorithms need to both "memorize" and "generalize". That is, for each label they need to memorize the type of queries that are relevant; e.g., for a product to product recommendation scenario, the algorithm should memorize which products can lead to click on a particular product using the provided product-product co-click data. At the same time, the algorithm should generalize on unseen queries.

It is a prevailing belief in the XMC community that due to the semantic gap and knowledge-intensive nature of XMC benchmarks, DE by themselves are not sufficient to attain good performance (Dahiya et al., 2023b). As a measure to overcome the semantic gap and enable memorization, SOTA XMC methods augment DE with either per-label classifiers (Dahiya et al., 2023a; Mittal et al., 2021a) or extra auxiliary parameters (Dahiya et al., 2023b). We explore this belief by performing a simple

experiment on a synthetic dataset where a random query text is associated with a random document text, and the task is to memorize these random correlations during retrieval. As shown in Section C.1, DE models are able to perform this task with perfect accuracy at least on up to 1M scale datasets, disputing the previously held belief in the literature.

In light of the aforementioned synthetic experiment, this work aims to answer the following question: "Are pure DE models sufficient for XMC tasks?" A Pure DE model is desirable because: 1). For XMC methods, unlike DE, the model size and consequently the number of trainable parameters scales *linearly* with the number of labels (see Figure 1). 2). XMC methods require re-training or model updates on encountering new labels. In contrast, DE methods can generalize to new labels based on their features. Interestingly, our work shows that pure DE models can indeed match or even outperform SOTA XMC methods by up to 2% even on the largest public XMC benchmarks while being 20× smaller in model size. The key to the improved performance is the right loss formulation for the underlying task and the use of extensive negatives to give consistent and unbiased loss feedback.

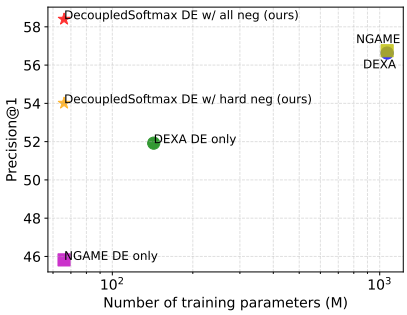

Figure 1: Number of trainable parameters used by different models and their P@1 performance on LF-AmazonTitles-1.3M dataset

Our proposed changes to DE training are based on the observation that XMC problems involve multiple positives per query and a long tail of labels, which at query level manifests as a large number of highly relevant negatives. Standard contrastive learning losses used to train DE models, e.g., InfoNCE loss (Oord et al., 2018), are not well-suited for multi-label learning tasks. Moreover, due to the computational requirements posed by encoding a large number of negative labels per query, standard approaches do not train DE models with sufficient number of negatives, whereas typical extreme classifiers are trained with significantly more ($\mathcal{O}(100)$) negatives per query. We provide a simple fix to InfoNCE loss to allow for multi-label correlation between query and labels by proposing a decoupled loss that decomposes the loss over individual positives. This is required since standard contrastive losses forces all positive labels to have scores in similar range but not all positive labels possess equal ease of prediction from the available data. For example, in EURLex-4K dataset labels "afghanistan" and "international_sanctions" often occur together but "afghanistan" is much easier to infer from the query text than "international_sanctions". In Section 4 we provide a more detailed motivation/justification for our proposed change to the InfoNCE loss. To establish the capabilities of DE methods, we first study our proposed loss using *all* the negatives in the loss term. Naturally, including all negatives is challenging for million-label scale datasets. To this end, we provide a memory-efficient distributed implementation that can use multiple GPUs and a modified gradient cache (Gao et al., 2021) approach to scale DE training to largest public XMC benchmarks even with a modest GPU setup (see Table 2, 9). We further study the performances and implications of approximations of the proposed loss functions using standard hard negative mining approaches. Since many practical settings focus on top-k ranking performance, we further refine our loss design to this end. Based on a soft *differentiable* version of top-k operation, we propose a novel loss function and show it can be integrated easily with standard DE training.

Before providing a detailed account of our contributions in Section 4 and 5, we begin by discussing relevant literature in Section 2 and introducing necessary background in Section 3.

2 Related work

Extreme classification methods. There is a large body of literature on utilizing sparse features or dense pre-trained dense features to develop XMC methods (Babbar & Schölkopf, 2017; Prabhu et al., 2018; Prabhu & Varma, 2014; Joulin et al., 2017; Yen et al., 2016; 2017; Gupta et al., 2021; Yu et al., 2022). Lately, task-specific dense features, e.g., based on BERT (Devlin et al., 2019), have been also utilized to design solutions for XMC problems (see, e.g., Zhang et al., 2018; You et al., 2019; Gupta et al., 2019; Guo et al., 2019; Medini et al., 2019; Chang et al., 2020; Jiang et al., 2021; Ye et al., 2020; Zhang et al., 2021; Gupta et al., 2022, and references therein). However, these approaches treat labels as featureless ids and employ classification networks that assign a unique classification

vector to each label. Given their increasingly popularity in information retrieval literature (Lee et al., 2019; Karpukhin et al., 2020; Luan et al., 2021; Qu et al., 2021; Xiong et al., 2021), DE methods that employ two encoders to embed query and items into a common space have been increasingly explored for XMC settings as well. By leveraging the label features such as label text (Mittal et al., 2021a; Dahiya et al., 2021), images (Mittal et al., 2022), and label (correlation) graph (Mittal et al., 2021b; Saini et al., 2021), such methods have already shown improved performance for the settings involving tail labels. That said, the current SOTA methods such as NGAME (Dahiya et al., 2023a), do not solely rely on a DE model. For instance, NGAME's two-stage approach involves first training query representations via a DE approach, and then utilizing a classification network in the second stage. DEXA (Dahiya et al., 2023b), is a recently proposed approach which builds on NGAME to improve the encoder embeddings by augmenting them with auxiliary parameters.

Loss functions for multi-label classification problems. One of the prevalent approaches to solve multi-label classification problem involves reducing it to multiple (independent) binary (Dembczyński et al., 2010) or multiclass (Joulin et al., 2017; Jernite et al., 2017) classification problems. Subsequently, one can rely on a wide range of loss functions designed for the resulting problems, aimed at minimizing the top-k classification error (Lapin et al., 2016; Berrada et al., 2018; Yang & Koyejo, 2020; Petersen et al., 2022). Since different queries have varying number of relevant documents, Lapin et al. (2016); Yang & Koyejo (2020) identified desirable loss functions that are simultaneously calibrated (Bartlett et al., 2006; Zhang, 2004) for multiple values of k. Another popular approach to solve a multi-label classification problem relies on (contextual) *learning to rank* framework (Liu et al., 2009) where (given a query) the main objective is to rank relevant documents above irrelevant ones. Various surrogate ranking loss functions, aimed at optimizing the specific ranking metrics such as Precision@k and NDCG@k, have been considered in the literature (see e.g., Usunier et al., 2009; Kar et al., 2015; Li et al., 2017; Jagerman et al., 2022; Su et al., 2022, and references therein).

Negative sampling. Modern retrieval systems increasingly deal with a large number of items, leading to increased training complexity. *Negative sampling* (Bengio & Senecal, 2008; Mikolov et al., 2013a; Mnih & Teh, 2012; Mnih & Kavukcuoglu, 2013) helps mitigate this via employing a loss function restricted to the given relevant documents and a subset of negatives or irrelevant documents. Selection of informative negatives is critical for training high-quality models. For classification networks, Bengio & Senecal (2008) studied negative sampling for softmax cross entropy loss by relying on importance sampling. Recently, many works have extended this line of work, e.g., by leveraging quadratic kernels (Blanc & Rendle, 2018) and random Fourier features (Rawat et al., 2019). As for the generic loss function and model architecture, it is common to employ sparse retrieval mechanism, e.g., BM25 (Robertson et al., 1995; 2004), to obtain negative documents (Luan et al., 2021; Karpukhin et al., 2020). Another strategy is to leverage *in-batch negatives* (Henderson et al., 2017; Karpukhin et al., 2020): for a given query in a mini-batch, items appearing as positive labels for other queries in the mini-batch are treated as negatives. However, in-batch negatives often constitute "easy" negatives (with very small contribution to the overall gradient) and do not aid the training much (see, e.g., Karpukhin et al., 2020; Luan et al., 2021). Same holds for the negatives sampled according to uniform or unigram distribution (Mikolov et al., 2013b). Reddi et al. (2019) proposed sampling a large uniform subset of items and weighting those according to their hardness for each query. Taking this approach to an extreme, for each query, ANCE (Xiong et al., 2021) selects the hardest negatives from the entire corpus according to the query-document similarity based on a document index.

3 BACKGROUND: MULTI-LABEL CLASSIFICATION

As discussed in Section 1, the (many-shot) retrieval problem explored in this work is essentially a multi-label classification problem. Assuming that $\mathcal{Q}$ and $\mathcal{D}$ denote the query and document (or label) spaces, respectively, the underlying *query-document relevance* is defined by the distribution P where $\mathsf{P}(d|q)$ captures the true relevance for the query-document pair $(q, d) \in \mathcal{Q} \times \mathcal{D}$. In this paper, we assume that the document space is finite with a total $|\mathcal{D}| = L$ documents. Let the training data consists of N examples $\mathcal{S} := \{(q_i, \mathbf{y}_i)\}_{i \in N} \subseteq \mathcal{Q} \times \{0, 1\}^L$, where, for $j \in [L]$, $y_{i,j} = 1$ *iff* jth document in $\mathcal{D}$ is relevant for query q_i. We also denote the set of positive labels for q_i as $P_i = \{j : y_{i,j} = 1\}$

Dual-encoder models and classification networks. Note that learning a retrieval model is equivalent to learning a *scoring function* (or simply *scorer*) $s_{\boldsymbol{\theta}} : \mathcal{Q} \times \mathcal{D} \to \mathbb{R}$, which is parameterized by model

parameters $\boldsymbol{\theta}$. A DE model consists of a *query encoder* $f_{\boldsymbol{\phi}} : \mathcal{Q} \to \mathbb{R}^d$ and a *document encoder* $h_{\boldsymbol{\psi}} : \mathcal{D} \to \mathbb{R}^d$ that map query and document features to d-dimensional embeddings, respectively. Accordingly, the corresponding scorer for the DE model is defined as: $s_{\boldsymbol{\theta}}(q, d) = \langle x_q, z_d \rangle = \langle f_{\boldsymbol{\phi}}(q), h_{\boldsymbol{\psi}}(d) \rangle$, where $\boldsymbol{\theta} = [\boldsymbol{\phi}; \boldsymbol{\psi}]$ and $x_q, z_d \in \mathbb{R}^d$. Note that it is common to share encoders for query and document, i.e., $\boldsymbol{\phi} = \boldsymbol{\psi}$. We also focus on this shared parameter setting in our exploration.

Different from DE models, classification networks ignore document features. Such networks consist of a query encoder $g_{\boldsymbol{\phi}} : \mathcal{Q} \to \mathbb{R}^d$ and a classification matrix $W \in \mathbb{R}^{L \times d}$, with ith row as the classification weight vector for the ith document. The scorer for the classification network then becomes $s_{\boldsymbol{\theta}}(q, d) = \langle g_{\boldsymbol{\phi}}(q), \mathbf{w}_d \rangle$, where $\mathbf{w}_d$ denotes the classification vector associated with document d and $\boldsymbol{\theta} = [\boldsymbol{\phi}; W]$. For ease of exposition, we do not always highlight the explicit dependence on trainable parameters $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\psi}$ while discussing DE models and classification networks.

Loss functions. Given the training set $\mathcal{S}$, one learns a scorer by minimizing the following objective:

$$\mathcal{L}(s; \mathcal{S}) = \frac{1}{N} \sum\nolimits_{i \in [N]} \ell(q_i, \mathbf{y}_i; s), \tag{1}$$

where ℓ is a *surrogate loss-function* for some specific metrics of interest (e.g., Precision@k and Recall@k). Informally, $\ell(q, \mathbf{y}; s)$ serves as a proxy of the quality of scorer s for query q, as measured by that metric with $\mathbf{y}$ as ground truth labels. One of the popular loss functions for multi-label classification problem is obtained by employing one-vs-all (OvA) reduction to converting the problem into L binary classification tasks (Dembczyński et al., 2010). Subsequently, one can invoke a binary classification loss function such as sigmoid *binary cross-entropy* (BCE) loss for L tasks, which leads to OvA-BCE:

$$\ell(q, \mathbf{y}, s) = -\sum\nolimits_{j \in [L]} \Big(y_j \cdot \log \frac{e^{s(q,d_j)}}{1 + e^{s(q,d_j)}} + (1 - y_j) \cdot \log \frac{1}{1 + e^{s(q,d_j)}} \Big). \tag{2}$$

Alternatively, one can employ a multi-label to multi-class reduction (Menon et al., 2019) and invoke softmax cross-entropy (SoftmaxCE) loss for each positive label:

$$\ell(q, \mathbf{y}, s) = -\sum\nolimits_{j \in [L]} y_j \cdot \log \Big(e^{s(q,d_j)} / \sum\nolimits_{l \in [L]} e^{s(q,d_l)} \Big). \tag{3}$$

4 Improved Training of Dual-Encoder Models

In this section, we first discuss the loss function typically used for training dual-encoders, and its shortcomings for multi-label problems. We then suggest simple fixes to address the problems, argue why these changes are necessary, and discuss practical implications of our modifications. Finally, we discuss our proposed soft top-k loss formulation and discuss its implementation.

4.1 Limitations of standard contrastive loss functions for extreme multi-label problems

Existing dual-encoder methods generally rely on contrastive losses for training. For the rest of the paper, we will anchor our technique, experiments, and discussion on the popular InfoNCE contrastive learning loss (Oord et al., 2018), but the observations and the approach apply to other contrastive learning loss functions considered in Xiong et al. (2021); Dahiya et al. (2021; 2023a). In a multi-label setting, given a batch of queries $\mathcal{B}$ and their positive labels, InfoNCE loss takes the form:

$$\mathcal{L}(s; \mathcal{B}) = \sum_{q_i \in \mathcal{B}} \ell(q_i, \mathbf{y}_i; s) = -\sum_{q_i \in \mathcal{B}} \sum_{j \in [L]} y_{ij} \cdot \log \frac{e^{s(q_i,d_j)}}{e^{s(q_i,d_j)} + \sum_{d^- \in \mathcal{D}^- \setminus d_j} e^{s(q_i,d^-)}}, \tag{4}$$

where $s(q_i, d_j)$ is the similarity score for query q_i and document d_j. $\mathcal{D}^-$ denotes the set of (hard) negatives created for the batch $\mathcal{B}$. That is, for in-batch negatives, $\mathcal{D}^-$ represents union of all positive labels for all the queries $q_i \in \mathcal{B}$. In the extreme case when we consider all labels in $\mathcal{D}^-$, (4) becomes the same as the SoftmaxCE loss in (3). A quick analysis of the gradients of this loss function reveals that:

$$\frac{\partial \ell_i}{\partial z_{d_l}} = \begin{cases} (1 - |P_i| \sigma_{il}) x_{q_i} & \text{if } l \in P_i \\ -|P_i| \sigma_{il} x_{q_i} & \text{otherwise} \end{cases}; \sigma_{ij} = \frac{e^{s(q_i,d_j)}}{\sum_k e^{s(q_i,d_k)}}, \text{where } s(q, d) = x_q^T z_d$$

This implies that the loss function is optimized when all positive labels score $\sigma_{il} = \frac{1}{|P_i|}$, while all negative labels achieve $\sigma_{il} = 0$. While this approach appears to be appropriate for elevating the ranking of positive labels above negatives, it raises concerns when applied to imbalanced real-world XMC datasets. In these datasets, not all positive labels possess equal ease of prediction from the available data. To illustrate this issue, consider a scenario where one positive label, denoted as l, is straightforward to predict for a given query q_i, while the remaining positive labels are characterized by ambiguity. Consequently, the model assigns a high score to the unambiguous label l. However, this leads to an increase in the softmax score for label l, resulting in σ_{il} exceeding $\frac{1}{|P_i|}$. Consequently, the loss function begins penalizing this particular positive label. Given the inherent label imbalance in XMC datasets, the uniform scoring of all positive labels, and the subsequent penalization of confidently predicted positives, do not align with the ideal modeling strategy.

Decoupled multi-label InfoNCE formulation. Based on the above observations, we re-formulate the multi-label InfoNCE loss as:

$$\text{(DecoupledSoftmax)} \quad \ell(q_i, \mathbf{y}_i; s) = -\sum_{j \in [L]} y_{ij} \cdot \log \frac{e^{s(q_i,d_j)}}{e^{s(q_i,d_j)} + \sum_{l \in [L]} (1 - y_{il}) \cdot e^{s(q_i,d_l)}} \tag{5}$$

where $s(q, d)$ is model assigned score to the query-document pair. Note that we *exclude* other positives from the softmax's denominator when computing the loss for a given positive, and add all the negatives in the denominator to eliminate the bias/variance in the estimation of that term. Analyzing the gradients with the proposed change gives:

$$\implies \frac{\partial \ell_i}{\partial z_{d_l}} = \begin{cases} (1 - \sigma_{il}) x_{q_i} & \text{if } l \in P_i \\ -\sum_{j \in P_i} \frac{e^{s(q_i,d_l)}}{e^{s(q_i,d_j)} + \sum_{k \notin P_i} e^{s(q_i,d_k)}} x_{q_i} & \text{otherwise} \end{cases}$$

$$\text{here } \sigma_{ij} = \frac{e^{s(q_i,d_j)}}{e^{s(q_i,d_j)} + \sum_{k \notin P_i} e^{s(q_i,d_k)}}, \text{ where } s(q, d) = x_q^T z_d$$

It is simple to see DecoupledSoftmax is optimized when we have $\sigma_{il} = 1$ if $l \in P_i$ (for positives) and when the score for a negative $s(q_i, d_l)$ is significantly smaller than all positive scores, i.e., $s(q_i, d_j)$ for $j \in P_i$. Thus, DecoupledSoftmax avoids penalizing positives that are confidently predicted by the model and allows the training to better handle the imbalanced multi-label nature of XMC datasets.

We empirically validate the efficacy of our proposed modification in addressing the aforementioned issues. To achieve this, we construct a synthetic dataset that accentuates the challenges and demonstrate how our modification mitigates these challenges, in contrast to its absence. Specifically, we create this synthetic dataset as follows: We randomly sample 1000 training query texts and 5000 label texts. For the initial 100 training queries, we intentionally designate the first token as "t*" (a randomly selected token) and associate all these queries with the first five labels i.e. labels 0, 1, 2, 3, and 4. However, for the 0th label, we append the token "t*" to its label text. This augmentation aims to facilitate the model's learning of the pattern that whenever "t*" is present in a query, label 0 should be confidently predicted. The test set comprises 1000 new, randomly generated query texts, wherein we again replace the first token with "t*" and exclusively tag these queries with the 0th label.

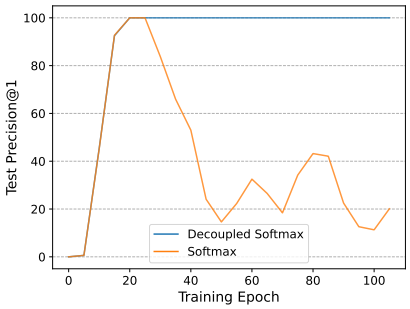

Figure 2: Decoupled softmax vs standard softmax on synthetic dataset.

As illustrated in Figure 2, our proposed modification results in a 100% $P@1$ on this dataset, while the standard softmax approach achieves approximately 20% $P@1$. This stark contrast highlights the softmax's inability to capture this straightforward correlation.

We note that similar situations arise in real-world XMC datasets. For instance in EURLex-4K dataset, labels such as 'afghanistan,' 'international_sanctions,' 'foreign_capital,' and 'cfsp' frequently co-occur in queries. However, 'afghanistan' is easily predictable, as training queries consistently contain the 'afghanistan' keyword, whereas the predictability of the other labels is not trivial. Thus,

the observations in Figure 2 are not limited to artificially crafted scenarios, but extend to practical XMC datasets, emphasizing the relevance of our proposed modification. In Figure 3 we visualize the gradients for two labels from the EURLex-4K dataset for all positive training queries encountered during training. This plot reveals that without modification the softmax loss gives very high variance gradient feedback, but with the change the gradient feedback is much more consistent and is always positive.

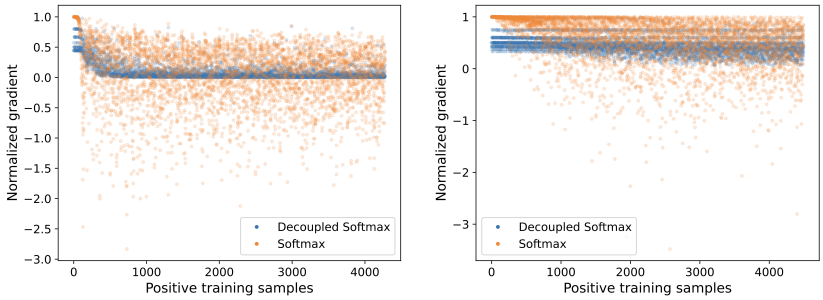

Figure 3: Gradient analysis of two labels [*left*] "afghanistan" (cherry-picked) and [*right*]"data_transmission" (randomly-chosen) on EURLex-4K dataset for all positive training queries encountered during training.

4.2 Memory Efficient Training

Training a DE model with the above loss function for a very large number of labels is challenging. As the pool of labels to be considered in the loss computation grows, the computational requirements to process these examples increase significantly – as for each query and label, the model needs to compute the embeddings and store all the intermediate activations to facilitate backpropagation. In order to overcome these memory constraints, we utilize a modified version of gradient cache approach (Gao et al., 2021) to perform distributed DE training with a large pool of labels under limited GPU memory. Due to space constraints, we describe our implementation and its runtime/memory analysis in detail in Section B.5 of the appendix.

Although, with the memory-efficient implementation we are able to train DE with the full loss even on the largest XMC benchmarks and establish the capabilities of DE models for XMC tasks with a modest GPU setup (see Table 9), the training is still expensive and is not desirable for scaling to even bigger datasets. To this end, we explore (see Section 5.4) approximations of our proposed approach with standard negative mining techniques similar to (Xiong et al., 2021) and evaluate the performance of the learned DE model with the increasing number of negative samples. Furthermore, in Section C.3, following the embedding cache approach proposed in (Lindgren et al., 2021), we also study how the quality of the sampled negatives affects the model performance.

4.3 Proposed differentiable top-k operator-based loss

Many-shot retrieval systems are applicable in a variety of fixed-size retrieval settings like say product-product retrieval – similar to LF-AmazonTitles-1.3M dataset – where the goal is to retrieve top-k products for a given input product. While our modification to InfoNCE loss does tend to give higher scores to positives than the negatives, it might be a bit too harsh in certain cases. For example, consider there are five positives and it is difficult to rank all the five positives over all the negatives. Suppose the goal is to optimize Recall@10. Then as long as the five positives are ranked above *all but five* negatives, the loss should be small (ideally zero). However, the proposed loss in (5) would still penalize such solutions.

To explicitly optimize such top-k metrics, we propose to use a differentiable soft top-k operator to define the loss function. The soft top-k operator takes a score vector as input, and essentially serves as a filter, assigning values close to 1 for the top-k scores and values close to 0 for the remaining scores. It is designed to be differentiable to allow gradients to flow through it during backpropagation.

Once we have the soft top-k score vector, we apply a log-likelihood loss over this vector. This loss serves to provide the model with feedback on its performance in placing all positives among the top-K predictions. When all positive labels are among the top-k predictions, the loss is zero. However,

if any positive label falls outside the top-k scores, the loss provides a non-zero feedback signal that guides the model to adjust its parameters during backpropagation. That is, given query q_i, let the score vector $\mathbf{s}_i := \big(s(q_i, d_1), \ldots, s(q_i, d_L)\big) \in \mathbb{R}^L$ and $\mathbf{z}_i := \mathsf{SoftTop\text{-}k}(\mathbf{s}_i)$ be the output of the soft top-k operator. With $\mathbf{y}_i \in \{0, 1\}^L$ as the label vector, the loss function for q_i takes the form:[1]

$$(\mathsf{SoftTop\text{-}k}) \quad \ell(q_i, \mathbf{y}_i; s) = -\frac{1}{L} \sum\nolimits_{j \in [L]} y_{ij} \cdot \log z_{ij}. \tag{6}$$

Since z_{ij} depends on the full score vector $\mathbf{s}_i$, this loss also provides feedback to all the positives and negatives in the dataset. To complete the loss function formulation, we specify the differentiable soft top-k operator below. Note that the operator is based on a proposal in (https://math.stackexchange.com/users/7072/thomas ahle).

Soft differentiable top-k formulation. We first look at a variational definition of hard top-k operator. Let $\mathbf{x} = \{x_i\}_{i=1}^N$ be the input vector, then $\mathsf{HardTop\text{-}k}(x) = \{s(x_i + t_\mathbf{x})\}_{i=1}^N$, where $s(a) = I[a > 0]$ is the step function that outputs 1 if $a > 0$ and 0, otherwise. $t_\mathbf{x}$ is a threshold so that exactly k elements exceed it, i.e., $\sum_{i=1}^N s(x_i + t_\mathbf{x}) = k$; for simplicity, let $\mathbf{x}$ comprise unique elements.

Based on HardTop-k operator, we can now define SoftTop-k operator by essentially replacing the above hard step function with a smooth σ function such as the standard sigmoid function. We can change the smoothness of our soft approximator by changing the σ function. We use a scaled sigmoid function i.e. $\sigma(x) = \text{sigmoid}(\alpha x)$ where $\alpha > 0$ is a hyperparameter that controls the smoothness. In general, we might not have a closed-form solution for $t_\mathbf{x}$ depending on our choice of σ. However, for monotonic σ, we can find $t_\mathbf{x}$ using binary search: start with an under- and over-estimate of $t_\mathbf{x}$ and at every iteration reduce the search space by half based on comparison of $\sum \sigma(x_i + t_\mathbf{x})$ and k.

Computing gradient of SoftTop-k operator. By definition, $\mathsf{SoftTop\text{-}k}(\mathbf{x}) = \{\sigma(x_i + t_\mathbf{x})\}_{i=1}^N$, where $t_\mathbf{x}$ is computed using binary search. Now,

$$\sum_j \sigma(x_j + t_\mathbf{x}) = k \implies \frac{\partial t_\mathbf{x}}{\partial x_i} = \frac{-\sigma'(x_i + t_\mathbf{x})}{\sum_j \sigma'(x_j + t_\mathbf{x})}.$$

$$\text{Hence, } \frac{\partial \mathsf{SoftTop\text{-}k}(\mathbf{x}_i)}{\partial x_l} = \sigma'(x_i + t_\mathbf{x})(\mathbb{1}_{[i=l]} + \frac{\partial t_\mathbf{x}}{\partial x_l}) = \sigma'(x_i + t_\mathbf{x})(\mathbb{1}_{[i=l]} - \frac{\sigma'(x_l + t_\mathbf{x})}{\sum_j \sigma'(x_j + t_\mathbf{x})}). \tag{7}$$

Combining definition of SoftTop-k operator along with associated loss function in (6), and by using gradient of SoftTop-k in (7), we now get an approach – SoftTop-k DE – to train DE models in extreme multi-label kind of scenarios. Note that here again we have to compute gradient wrt all negatives; thus, requiring the memory efficient and distributed backpropagation implementation from Section 4.2.

5 Experiments

In this section we compare DE models trained with the proposed loss functions to XMC methods. We show that the proposed loss functions indeed substantially improve the performance of DE models, even improving over SOTA in some settings. We also present ablations validating choices such as loss functions, number of negatives, and further provide a more exhaustive analysis in Section C of the Appendix

5.1 Datasets and evaluation

We experiment with diverse XMC datasets, EURLex-4K, LF-AmazonTitles-131K, LF-Wikipedia-500K, and LF-AmazonTitles-1.3M, each containing label texts. These datasets cover various applications like product recommendation, Wikipedia category prediction, and EU-law document tagging. We follow standard setup guidelines from the XMC repository (Bhatia et al., 2016) for

Table 1: Dataset statistics

Dataset	Num Train Points	Num Test Points	Num Labels	Avg Labels per Point	Avg Points per Label
EURLex-4K	15,449	3,865	3801	5.30	20.79
LF-AmazonTitles-131K	294,805	134,835	131,073	2.29	5.15
LF-Wikipedia-500K	1,779,881	769,421	501,070	4.75	16.86
LF-AmazonTitles-1.3M	2,248,619	970,237	1,305,265	22.20	38.24

[1]Note that, with slight abuse of notation, we use $\mathsf{SoftTop\text{-}k}(\cdot)$ and SoftTop-k to refer to a soft version of the top-k operator and the corresponding loss function, respectively.

Table 2: Performance comparison on large XMC benchmark datasets. Methods suffixed with $\times 3$ use an ensemble of 3 learners. Bold numbers represent overall best numbers while underlined numbers represent best numbers among pure DE methods. Results for most existing XMC baselines are from either their respective papers or the XMC repository (Bhatia et al., 2016), blank entries indicate source does not have those numbers.

Method	Label Text	Label Classifier	P@1	P@5	N@5	PSP@5	P@1	P@5	N@5	PSP@5
			LF-Wikipedia-500K				LF-AmazonTitles-1.3M			
DistilBERT OvA Classifier	✗	✓	82.00	48.54	73.44	48.87	48.72	39.09	44.79	25.91
ELIAS	✗	✓	81.96	48.90	73.71	48.92	49.26	39.29	45.44	27.37
XR-Transformer $\times 3$	✗	✓	81.62	47.85	72.43	47.81	50.14	39.98	46.59	27.79
ECLARE $\times 3$	✓	✓	68.04	35.74	56.37	38.29	50.14	40.00	46.68	30.56
SiameseXML $\times 3$	✓	✓	67.26	33.73	54.29	37.07	49.02	38.52	45.15	32.52
NGAME	✓	✓	84.01	49.97	75.97	57.04	56.75	44.09	52.41	35.36
NGAME DE	✓	✗	77.92	40.95	-	57.33	45.82	35.48	-	36.80
DEXA	✓	✓	84.92	50.52	76.80	58.33	56.63	43.90	52.37	34.86
DEXA DE	✓	✗	79.99	42.52	-	57.68	51.92	38.86	-	**37.31**
DecoupledSoftmax DE	✓	✗	**85.78**	50.53	**77.11**	58.97	**58.40**	**45.46**	**54.30**	36.58
SoftTop-5 DE	✓	✗	82.93	**51.11**	76.61	**59.55**	54.41	43.82	51.70	32.20

LF-* datasets. For EURLex-4K, we adopt the XR-Transformer setup due to unavailable raw texts in the XMC repository version. Dataset statistics are in Table 1, and a summary is in Table 6. Model evaluation employs standard XMC and retrieval metrics (P@1, P@5, PSP@5, nDCG@5, R@10, R@100). Details are in Section B.4.

5.2 Baselines and setup

We compare our approach with SOTA XMC methods such as DEXA (Dahiya et al., 2023b), NGAME (Dahiya et al., 2023a), XR-Transformer (Zhang et al., 2021), ELIAS (Gupta et al., 2022), ECLARE (Mittal et al., 2021b), SiameseXML (Dahiya et al., 2021). To keep the comparison fair, we use the same 66M parameter `distilbert-base` transformer (Sanh et al., 2019) used in NGAME in all of our dual-encoder experiments and share the encoder parameters for both query and document. We also compare against encoder only version of NGAME and DEXA (represented by NGAME DE and DEXA DE), i.e., retrieval using only encoder representation. In order to have a direct comparison between a pure classification model trained with all negatives and our proposed approach, similar to the BERT-OvA approach in Gupta et al. (2022, Table 2), we also report results for the Distil-BERT OvA classifier model which stacks a classification layer with L labels on top of a `distilbert-base` transformer and fine-tunes both the classification layer and the transformer by using the full one-vs-all BCE loss in (2). See Appendix B for more details on the experimental setup.

5.3 Comparison with XMC methods

Table 2 presents the main comparison results, showcasing the performance of our approach relative to existing methods. On large-scale datasets such as LF-Wikipedia-500K and LF-AmazonTitles-1.3M, our approach significantly outperforms all existing XMC methods. More specifically, it can be upto 2% better than the next best method on P@K and up to 2.5% better on PSP@k, while being $20\times$ smaller (in terms of trainable parameters) than existing state-of-the-art method on LF-AmazonTitles-1.3M (cf. Figure 1). Moreover, compared to the DE versions of NGAME and DEXA our approach can be 6% better at P@k.

On smaller datasets such as EURLex-4K, XR-Transformer outperforms our approach but as we demonstrate in Section C of the appendix, XR-Transformer gains are mainly because it uses an ensemble of 3 learners and sparse classifier-based re-ranker. Such modifications can easily be incorporated into any of the methods. On LF-AmazonTitles-131K, our approach is comparable to DEXA/NGAME on P@5 but there is a considerable gap on P@1, which can be partially explained by a significant jump in performance at P@1 achieved by using a label propensity aware score fusion module in NGAME; see Dahiya et al. (2023a, Table 8).

Table 3: Performance comparison on small extreme classification datasets. Please refer to Table 2 for notations.

Method	Label Text	Label Classifier	P@1	P@5	N@5	PSP@5	P@1	P@5	N@5	PSP@5
			EURLex-4K				LF-AmazonTitles-131K			
DistilBERT OvA Classifier	✗	✓	85.25	59.96	69.83	53.72	37.55	18.49	40.75	40.18
ELIAS	✗	✓	86.88	61.73	**71.65**	55.62	39.26	19.02	42.23	41.89
XR-Transformer×3	✗	✓	**88.41**	**63.18**	-	-	38.10	18.32	40.71	39.59
NGAME	✓	✓	-	-	-	-	46.01	21.47	48.67	49.43
NGAME DE	✓	✗	-	-	-	-	42.61	20.69	-	48.71
DEXA	✓	✓	-	-	-	-	**46.42**	**21.59**	**49.00**	**49.65**
DEXA DE	✓	✗	-	-	-	-	44.76	21.18	-	49.50
DecoupledSoftmax DE	✓	✗	86.78	60.19	70.37	54.78	42.52	20.64	46.33	47.40
SoftTop-5 DE	✓	✗	83.42	60.95	70.07	**56.84**	41.11	21.23	46.58	48.70

5.4 DecoupledSoftmax WITH HARD NEGATIVES

Since our proposed DecoupledSoftmax loss uses all negatives in its formulation, we investigate the approximations of this loss formulation by considering only a few negatives per training query in the batch; see Table 4.

Following Xiong et al. (2021), for mining the hard negatives we compute the top 100 negative shortlist at regular intervals and sample the negatives from this shortlist. For each query we train on the full pool of documents sampled in the mini-batch; as a result, for a mini-batch of size B and number of sampled per query m, we compute the loss on almost $B(1+m)$ documents. Clearly from Table 4, increasing the number of effective labels considered in the loss significantly increases the performance of the approximate versions. In Section C.2.3 of the appendix we also compare the results of InfoNCE loss with hard negatives and DecoupledSoftmax loss with hard negatives.

Table 4: Ablation of number of negatives sampled per query in the DecoupledSoftmax DE training.

Batch size	Hard neg per query	Effective doc pool	P@1	P@5	R@10	R@100
		LF-AmazonTitles-1.3M				
8192	0	∼ 8192	42.15	32.97	29.28	57.48
8192	1	∼ 16384	49.16	39.07	32.76	60.09
8192	2	∼ 24576	50.74	40.14	33.31	60.45
8192	5	∼ 49152	52.04	40.74	33.48	60.13
8192	10	∼ 90112	54.01	42.08	34.19	61.04
		LF-Wikipedia-500K				
2048	0	∼ 2048	77.71	43.32	69.24	88.12
2048	1	∼ 4096	82.85	48.84	74.47	90.08
2048	2	∼ 6144	83.34	49.32	74.73	89.74
2048	5	∼ 12288	83.86	49.57	74.60	89.12
2048	10	∼ 22528	84.77	50.31	75.52	90.29

5.5 COMPARISON ACROSS DIFFERENT LOSS FUNCTIONS

Next, we compare different loss formulations as discussed in Sections 3 and 4 on the EURLex-4K and LF-AmazonTitles-131K datasets (cf. Table 5). The DE model fails to train with BCE loss in (2). We hypothesize that this might be due to the stringent demand of the BCE loss, which requires all negative pairs to exhibit high negative similarity and all positive pairs to have a high positive similarity – a feat that can be challenging for DE representations. DecoupledSoftmax loss in (5) performs the best for top-1 predictions. Moreover, the SoftTop-5 and SoftTop-100 losses (cf. (6)) yield optimal results for top-5 and top-100 predictions, respectively. These results show that SoftTop-k loss can outperform other loss formulations when optimizing for a specific prediction set size. We further analyze the label score distributions of models learned with different loss functions in Section C of the Appendix.

Table 5: Ablation of loss functions for OvA DE method.

Loss	P@1	P@5	R@10	R@100
	EURLex-4K			
BCE (2)	0.1	0.07	0.14	1.84
SoftmaxCE (3)	80.05	58.36	72.41	92.57
DecoupledSoftmax (5)	**86.78**	60.19	72.56	91.75
SoftTop-5 (6)	83.42	**60.95**	**74.21**	91.30
SoftTop-100 (6)	52.34	37.41	56.97	**93.72**
	LF-AmazonTitles-131K			
BCE (2)	12.75	5.96	17.18	25.20
SoftmaxCE (3)	41.77	20.87	56.91	69.49
DecoupledSoftmax (5)	**42.52**	20.64	56.36	68.52
SoftTop-5 (6)	41.11	**21.23**	**57.72**	69.46
SoftTop-100 (6)	33.38	18.27	53.87	**71.52**

Limitations. Our key contribution is a simple yet effective training procedure for DE methods that allows them to match or surpass SOTA extreme classification methods on standard benchmarks for many-shot retrieval. One limitation of our empirical evaluation is that we have only evaluated our proposal up to O(million)-sized label spaces. Verifying our gains on O(billion) scale is an interesting and challenging avenue for future work. Additionally, designing algorithms that can improve DE performance on XMC tasks without requiring all the negatives is another interesting direction. We investigated a well known many-shot learning problem and studied it in abstract form, so we don't envision significant additional negative implications for society.

REFERENCES

Rohit Babbar and Bernhard Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pp. 721–729, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346757. doi: 10.1145/3018661.3018741. URL https://doi.org/10.1145/3018661.3018741.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Yoshua Bengio and Jean-SÉbastien Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008. doi: 10.1109/TNN.2007.912312.

Leonard Berrada, Andrew Zisserman, and M. Pawan Kumar. Smooth loss functions for deep top-k classification. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hk5elxbRW.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL http://manikvarma.org/downloads/XC/XMLRepository.html.

Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 590–599. PMLR, 10–15 Jul 2018.

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 and Data Mining*, KDD '20, pp. 3163–3171, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403368. URL https://doi.org/10.1145/3394486.3403368.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pp. 191–198, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959190. URL https://doi.org/10.1145/2959100.2959190.

K. Dahiya, N. Gupta, D. Saini, A. Soni, Y. Wang, K. Dave, J. Jiao, K. Gururaj, P. Dey, A. Singh, D. Hada, V. Jain, B. Paliwal, A. Mittal, S. Mehta, R. Ramjee, S. Agarwal, P. Kar, and M. Varma. Ngame: Negative mining-aware mini-batching for extreme classification. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, March 2023a.

K. Dahiya, S. Yadav, S. Sondhi, D. Saini, S. Mehta, J. Jiao, S. Agarwal, P. Kar, and M. Varma. Deep encoders with auxiliary parameters for extreme classification. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2023b.

Kunal Dahiya, Ananye Agarwal, Deepak Saini, Gururaj K, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2330–2340. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/dahiya21a.html.

Krzysztof Dembczyński, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 279–286, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pp. 316–321, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.repl4nlp-1.31. URL https://aclanthology.org/2021.repl4nlp-1.31.

Chuan Guo, Ali Mousavi, Xiang Wu, Daniel Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. *Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/1908.10396.

N. Gupta, S. Bohra, Y. Prabhu, S. Purohit, and M. Varma. Generalized zero-shot extreme multi-label learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2021.

N. Gupta, P.H. Chen, H-F. Yu, C-J. Hsieh, and I. Dhillon. Elias: End-to-end learning to index and search in large output spaces. In *Neural Information Processing Systems*, November 2022.

Vivek Gupta, Rahul Wadbude, Nagarajan Natarajan, Harish Karnick, Prateek Jain, and Piyush Rai. Distributional semantics meets multi-label learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3747–3754, 2019.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017.

Thomas Ahle (https://math.stackexchange.com/users/7072/thomas ahle). Differentiable top-k function. Mathematics Stack Exchange. URL https://math.stackexchange.com/q/4506773. URL:https://math.stackexchange.com/q/4506773 (version: 2022-08-07).

Rolf Jagerman, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. On optimizing top-k metrics for neural ranking models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pp. 2303–2307, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531849. URL https://doi.org/10.1145/3477495.3531849.

Yacine Jernite, Anna Choromanska, and David Sontag. Simultaneous learning of trees and representations for extreme classification and density estimation. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1665–1674. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/jernite17a.html.

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.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431, Valencia, Spain, April

2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2068.

Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 189–198, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/kar15.html.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550.

Maksim Lapin, Matthias Hein, and Bernt Schiele. Loss functions for top-k error: Analysis and insights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1468–1477, 2016.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 6086–6096. Association for Computational Linguistics, 2019.

Yuncheng Li, Yale Song, and Jiebo Luo. Improving pairwise ranking for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Erik Lindgren, Sashank Reddi, Ruiqi Guo, and Sanjiv Kumar. Efficient training of retrieval models using negative cache. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4134–4146. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2175f8c5cd9604f6b1e576b252d4c86e-Paper.pdf.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021. doi: 10.1162/tacl_a_00369. URL https://aclanthology.org/2021.tacl-1.20.

Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. *Advances in Neural Information Processing Systems*, 32, 2019.

Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Multilabel reductions: what is my loss optimising? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/da647c549dde572c2c5edc4f5bef039c-Paper.pdf.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013b. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, 2018. ISSN 1554-0669. doi: 10.1561/1500000061. URL http://dx.doi.org/10.1561/1500000061.

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*, WSDM '21, pp. 49–57, New York, NY, USA, 2021a. Association for Computing Machinery. ISBN 9781450382977. doi: 10.1145/3437963.3441807. URL https://doi.org/10.1145/3437963.3441807.

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*, WWW '21, pp. 3721–3732, New York, NY, USA, 2021b. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449815. URL https://doi.org/10.1145/3442381.3449815.

Anshul Mittal, Kunal Dahiya, Shreya Malani, Janani Ramaswamy, Seba Kuruvilla, Jitendra Ajmera, Keng-Hao Chang, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Multi-modal extreme classification. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12383–12392, 2022. doi: 10.1109/CVPR52688.2022.01207.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/db2b4182156b2f1f817860ac9f409ad7-Paper.pdf.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 419–426, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Felix Petersen, Hilde Kuehne, Christian Borgelt, and Oliver Deussen. Differentiable top-k classification learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17656–17668. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/petersen22a.html.

Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 263–272, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623651. URL https://doi.org/10.1145/2623330.2623651.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pp. 993–1002, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi: 10.1145/3178876.3185998. URL https://doi.org/10.1145/3178876.3185998.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5835–5847, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.466. URL https://aclanthology.org/2021.naacl-main.466.

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/e43739bba7cdb577e9e3e4e42447f5a5-Paper.pdf.

Sashank J. Reddi, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. Stochastic negative mining for learning with large output spaces. In Kamalika Chaudhuri and Masashi Sugiyama (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1940–1949. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/reddi19a.html.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pp. 42–49, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138741. doi: 10.1145/1031171.1031181. URL https://doi.org/10.1145/1031171.1031181.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. Galaxc: Graph neural networks with labelwise attention for extreme classification. In *Proceedings of the Web Conference 2021*, WWW '21, pp. 3733–3744, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449937. URL https://doi.org/10.1145/3442381.3449937.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Jianlin Su, Mingren Zhu, Ahmed Murtadha, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Zlpr: A novel loss for multi-label classification. *arXiv preprint arXiv:2208.02955*, 2022.

Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1057–1064, 2009.

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*, 2021. URL https://openreview.net/forum?id=zeFrfgyZln.

Forest Yang and Sanmi Koyejo. On the consistency of top-k surrogate losses. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10727–10735. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/yang20f.html.

Hui Ye, Zhiyu Chen, Da-Han Wang, and Brian D. Davison. Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Ian E. H. Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S. Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 3069–3077. JMLR.org, 2016.

Ian E.H. Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*,

KDD '17, pp. 545–553, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098083. URL https://doi.org/10.1145/3097983.3098083.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 32, 2019.

H-F. Yu, K. Zhong, J. Zhang, W-C. Chang, and I. S. Dhillon. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 2022.

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.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1), feb 2019. ISSN 0360-0300. doi: 10.1145/3285029. URL https://doi.org/10.1145/3285029.

Tong Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5(Oct):1225–1251, 2004.

Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, pp. 100–107, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450350464. doi: 10.1145/3206025.3206030. URL https://doi.org/10.1145/3206025.3206030.

Table 6: A snapshot of the datasets used in our experiments. Each row represents a different dataset with a sample query and associated labels/documents separated by ";"

Dataset	Query	Labels/Documents
EURLex-4K	"commiss decis octob aid scheme implement kingdom spain airlin intermediación aérea sl notifi document number spanish text authent..."	air_transport; airline; catalonia control_of_state_aid; invitation_to_tender; services_of_general_interest; spain; state_aidtransport_company
LF-AmazonTitles-131K	"Snowboard Kids 2"	Super Mario 64; Super Smash Bros.; Nintendo 64 System - Video Game Console; Donkey Kong 64; Diddy Kong Racing; Kirby 64: The Crystal Shards
LF-Wikipedia-500K	"bill slavick is an american retired professor and peace activist who ran for the u s senate in maine as an independent politician independent in the maine..."	1928 births; american roman catholics; living people; louisiana state university faculty; maine independents; politicians from portland maine; university of notre dame alumni
LF-AmazonTitles-1.3M	"Power Crunch 12-1.4 oz Cookies Peanut Butter Fudge Energy Bars BNRG"	RiseBar Protein Almond Honey, 2.1 oz. Bars, 12-Count; thinkThin Protein Bars, Caramel Fudge, Gluten Free, 2.1-Ounce Bars (Pack of 10); Vitafusion Fiber Gummies Weight Management, 90-Count Bottle; Kirkland Signature Extra Fancy Unsalted Mixed Nuts 2.5 (LB); 24 Detour Bars, Low Sugar Whey Protein Bars, 1.5oz Bars; BioNutritional Research Group Power Crunch Protein Triple Ch; Dymatize ISO100 Hydrolyzed 100% Whey Protein Isolate Gourmet Vanilla - 5 lbs; thinkThin Chocolate Espresso Gluten Free, 2.1-Ounce Bars (Pack of 10)...

A Future Work and Reproducibility

While our work shows that dual encoder model can be better or as competent as SOTA extreme classification methods without needing to linearly scale the model parameters with the number of labels, there are still challenges that need to be addressed. Below we highlight a few key future work directions:

- **Scaling to larger label spaces:** We have established the performance of small dual encoders on up to O(million) label spaces. However, it would be interesting to study up to how big datasets can the limited-size dual encoders can retain their performance and if there is any fundamental capacity threshold that is essential for scaling to practical web-scale datasets.
- **Improving DE performance without all negatives:** We have observed a clear relationship between DE performance and the availability of negatives. While it is computationally challenging to have all negatives, designing algorithms that can boost DE performance on many-shot retrieval problems without requiring a large pool of documents in loss computation is another promising area for exploration.
- **DE encoder with in-built fast search procedure:** DE approaches rely on external approximate nearest neighbor search routines to perform fast inference when searching for top documents for a given query. In contrast, many hierarchical extreme classification methods come in-built with fast search procedures, it would be interesting to explore such hierarchical approaches for DE models.

Reproducibility: Our `PyTorch` based implementation is available at the following link. We plan to open-source the code after the review period along with the trained models to facilitate further research in these directions. Below we provide additional details to clarify our approach and experimental setup.

B Implementation Details

B.1 Dataset

In this subsection we provide more details of the datasets considered in this work,

- **EURLex-4K**: The EURLex-4K dataset is a small extreme classification benchmark dataset. It was sourced from the publications office of the European Union (EU). Specifically, it's a collection of documents concerning European Union law. The dataset comprises approximately 4,000 distinct labels, represented by EuroVoc descriptors. EuroVoc is a multilingual thesaurus maintained by the EU, covering fields of EU interest and offering a consistent set of terms for the categorization of documents. There are over 15,000 training examples in the dataset. Each document within the EURLex-4K dataset is provided as a text string, consisting of the title and the content (or body) of the document.
- **LF-AmazonTitles-131K**: Originated from Amazon, this dataset encapsulates product relationships, aiming to predict associated products based on a queried product. It comprises about 131,000 labels (products for recommendation from the catalog), and around 300,000

training query products. The products are recognized via their respective Amazon product titles, which function as the principal descriptive feature.

- **LF-Wikipedia-500K**: A conventional large-scale extreme classification benchmark, this dataset comprises document tagging of Wikipedia articles. Human annotators have tagged query documents with multiple categories, amounting to a total of 500,000 labels and approximately 1.8 million training queries. It is marked by the presence of certain super-head classes.
- **LF-AmazonTitles-1.3M**: This dataset is similar to LF-AmazonTitles-131K, albeit on a larger scale and with a denser network. It includes a total of 2.2 million training points and 1.3 million labels. The distribution of these co-purchasing links can be significantly skewed. Some popular products relate to a considerable number of other products, while less popular ones maintain fewer connections.

B.2 Training hyperparameters

In all of our experiments we train the dual encoders for 100 epochs with `AdamW` optimizer and use the linear decay with warmup learning rate schedule. Following the standard practices, we use 0 weight deacy for all non-gain parameters (such as layernorm, bias parameters) and 0.01 weight decay for all the rest of the model parameters. Rest of the hyperparameters considered in our experiments are described below:

- `max_len`: maximum length of the input to the transformer encoder, similar to (Dahiya et al., 2023a) we use 128 `max_len` for the long-text datasets (EURLex-4K and LF-Wikipedia-500K) and 32 `max_len` for short-text datasets (LF-AmazonTitles-131K and LF-AmazonTitles-1.3M)
- `LR`: learning rate of the model
- `batch_size`: batch-size of the mini-batches used during training
- `dropout`: dropout applied to the dual-encoder embeddings during training
- α: multiplicative factor to control steepness of σ function described in Section 4.3
- η: micro-batch size hyperparameter that controls how many labels get processed at a time when using gradient caching
- τ: temperature hyperparameter used to scale similarity values (i.e. $s(q_i, d_j)$) during loss computation

Table 7: Hyperparameters

Dataset	**max_len**	**LR**	**batch_size**	**dropout**	α	η	τ
EURLex-4K	128	0.0002	1024	0.1	2	4096	0.05
LF-AmazonTitles-131K	32	0.0004	4096	0	2	2048	0.05
LF-Wikipedia-500K	128	0.0004	4096	0.1	2	2048	1
LF-AmazonTitles-1.3M	32	0.0008	8192	0.1	2	1024	1

B.3 Baselines and Evaluation Metrics

For existing extreme classification baselines (apart from ELIAS), we obtain their numbers from their respective papers or the extreme classification repository (Bhatia et al., 2016). Since the ELIAS paper do not report results on most of the datasets considered in this work, we ran the publicly available ELIAS implementation to report the results.

B.4 Evaluation Metrics

We evaluate the models using standard XMC and retrieval metrics such as P@1, P@5, PSP@5, nDCG@5, R@10, and R@100. P@k (*Precision* at k) measures the proportion of the top-k predicted labels that are also among the true labels for a given instance. PSP@k (*Propensity Scored Precision*

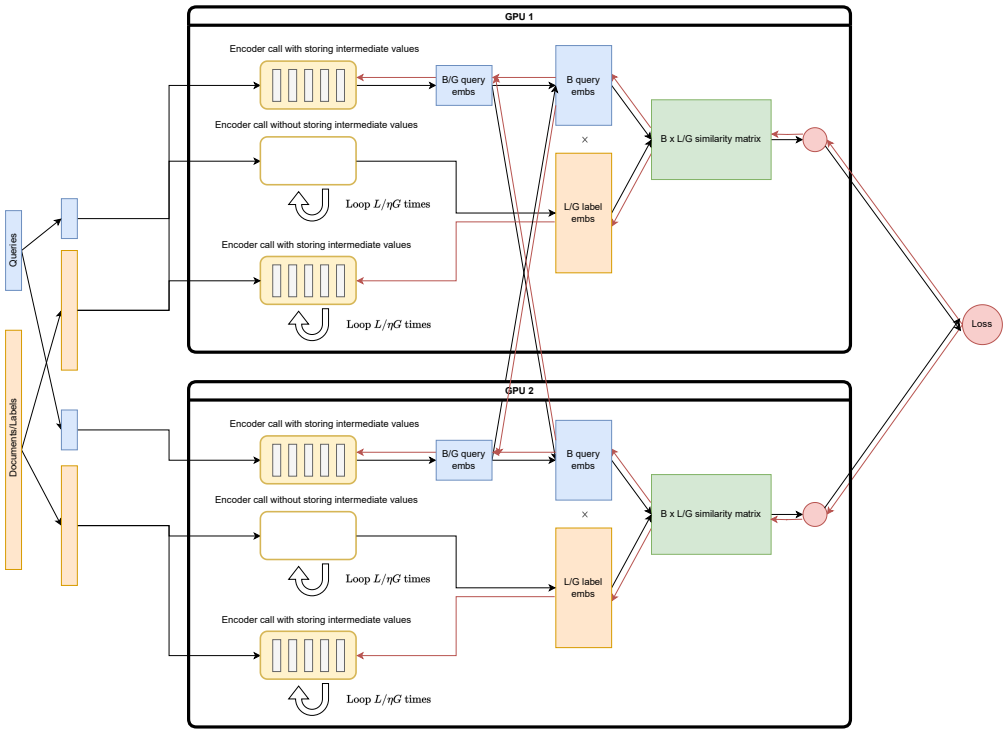

Figure 4: Illustration of distributed implementation with gradient caching applied on label embedding computation. Here solid black line indicate forward pass direction and solid red lines indicate gradient backpropagation direction. L is the number of labels considered in the loss computation, B is the batch size of queries, η is a micro batch-size hyperparameter which controls how many labels are processed at a time.

at k) is a variant of precision that takes into account the propensity scores of the labels, which are indicative of the likelihood of a label being positive. nDCG@k (*normalized Discounted Cumulative Gain* at k) measures the quality of the ranked list of labels up to the position k. R@k (*Recall* at k) is the proportion of the true labels that are included in the top-k predicted labels.

B.5 Memory efficient Distributed implementation

In this section we describe our distributed implementation in more detail. For a multi-GPU setting, where we have G GPUs, a batch size of B queries, and a pool size of L labels considered in the loss computation.

We begin by distributing the data and labels evenly across the GPUs. Each GPU receives B/G queries and L/G labels. However, for extreme multi-label problems with $>$ 1M labels, even this reduced load can be too large for the L/G labels – memory-wise – for an individual GPU during backpropagation. To handle this, we use the *gradient caching* method (Gao et al., 2021). More concretely, on each GPU, we first divide the large L/G label input batch into manageable sub-batches of size η. Then the encoder performs $L/\eta G$ forward passes to compute label embeddings without constructing the computation graph (i.e. `with torch.no_grad()`). Then, the loss is computed based on these embeddings, and the gradients for each embedding are computed and stored in a cache. Once we get the gradients with respect to the label embeddings, the encoder is run again on each η sized sub-batch separately (but this time we retain the computation graph), and the stored embedding gradients are used to backpropagate through the encoder parameters. This allows for constant memory usage during the encoder's forward-backward pass at the cost of doing the forward pass twice.

To compute the loss, we perform an *all-gather* operation on query embeddings across all GPUs. This enables each GPU to compute the dot product similarity matrix for the full B queries and L/G labels. The computation of the loss requires certain common terms, such as the denominator in the case of softmax-style losses. To compute these common terms, we perform an *all-reduce* operation across all

GPUs. Finally, each GPU computes the gradients with respect to its batch of L/G label and B/G query embeddings and backpropagates the respective gradients through the encoder parameters. By computing the loss and gradients in this distributed manner and by employing gradient cache, we can effectively handle large pools of labels without overwhelming the memory capacity of individual GPUs.

In the following description we assume that the loss is standard softmax loss but appropriate modifications can be done to make it work for other losses discussed in Section 4. Figure 4 illustrates the overall processing of a mini-batch in a 2 GPU setup. More generally, let there be G GPUs, B queries $\{x_i\}_{i=1}^{B}$ and L labels $\{l_i\}_{i=1}^{L}$ considered in the loss computation.

- Each GPU receives B/G queries and L/G labels. More concretely, we define the set of queries received by the g^{th} GPU as $\{x_i^g\}_{i=1}^{B/G}$ and the set of labels as $\{l_i^g\}_{i=1}^{L/G}$.
- We can then define the embedding matrices for each GPU for B/G queries and L/G labels as $Q_g = \phi(\{x_i^g\}_{i=1}^{B/G})$ and $L_g = \phi(\{l_i^g\}_{i=1}^{L/G})$ respectively, where ϕ is the query/document encoder. Note that, computing L_g for large L requires gradient caching as described in Section 4.2.
- We then gather all query embeddings at each GPU, denoted as $\hat{Q} = \text{Gather}(Q_g)_{g=1}^{G}$
- We compute the similarity matrix for B queries and L/G labels as $S_g = \hat{Q}L_g^T$.
- Finally, we compute the softmax loss for each query-label pair (i, j) (on respective GPUs) as $\ell_{i,j} = -(S_{i,j} - \texttt{logsumexp}_{j=1}^{L}(S_{i,j}))$. Note, since $S = [S_1, S_2, ..., S_G]$ is distributed across GPU, we will have to perform an ***all_reduce*** operation to compute $\texttt{logsumexp}_{j=1}^{L}(S_{i,j})$ term in the loss.

B.5.1 Memory and Runtime Analysis of Proposed implementation

Below we provide analysis of memory and compute requirements of the implementation with or without gradient cache based approach: let's assume we have G gpus, our batch size is B, number of labels L, gradient cache sub-batch size η, embedding dimension d and the memory required for processing (combined forward and backward pass of transformer encoder during training) a single query and label is M_q^{enc} and M_l^{enc} respectively. Without gradient caching, the total memory requirements for processing the whole mini-batch during training will be: $\mathcal{O}(\frac{B(M_q^{enc}+d)+L(M_l^{enc}+d)+BL}{G})$, notice that with transformer encoders LM_l^{enc} is the largest bottleneck for XMC datasets with L in order of millions. With gradient caching the memory requirement is: $\mathcal{O}(\frac{B(M_q^{enc}+d)+Ld+\eta M_l^{enc}+BL}{G})$. Similarly for compute let's assume the compute required for the forward and backward pass of query be T_q^{enc-f} and T_q^{enc-b} respectively, the compute required for the forward and backward pass of a label be T_l^{enc-f} and T_l^{enc-b}. Then, without gradient caching the total compute requirement is: $\mathcal{O}(\frac{B(T_q^{enc-f}+T_q^{enc-b}+d)+L(T_l^{enc-f}+T_l^{enc-b}+d)+BL}{G})$. With gradient caching the compute requirement will be: $\mathcal{O}(\frac{B(T_q^{enc-f}+T_q^{enc-b}+d)+L(2T_l^{enc-f}+T_l^{enc-b}+d)+BL}{G})$.

Empirically even with 16 A100 GPUs, the naive approach without gradient caching runs easily out of GPU memory even on LF-AmazonTitles-131K dataset as it demands about $\sim$TB combined GPU memory just for encoding all the labels. Table 8 provides the ablation of memory used and runtimes of the memory-efficient approach with different global batch sizes and gradient cache sub-batch sizes on LF-AmazonTitles-131K dataset (experiments ran with 4 A100 GPU setup):

B.6 Hardware and Runtimes

We run our experiments on maximum 16 A100 GPU setup each having 40 GB GPU memory. In principle, our experiments can also be performed on a single GPU setup, but it'll take significantly longer to train. Table 9 reports the training times and number of GPU used during training for each dataset.

Table 8: Ablation of memory used and runtimes of the grad-cache based memory efficient implementation

Global batch size	Gradient cache sub-batch size	Max Memory per GPU	Avg time per batch	Avg time per epoch
1024	64	6.03 GB	17.8s	1.42 hrs
1024	256	6.31 GB	7.1s	0.56 hrs
1024	1024	14.89 GB	5.9s	0.47 hrs
1024	4096	48.51 GB	5.6s	0.44 hrs
4096	256	15.75 GB	7.9s	0.16 hrs
4096	1024	16.16 GB	6.7s	0.14 hrs
4096	4096	51.85 GB	6.0s	0.12 hrs
16384	256	53.12 GB	12.2s	0.06 hrs
16384	1024	53.12 GB	8.2s	0.04 hrs
16384	4096	58.77 GB	6.6s	0.03 hrs

Table 9: Training times and resources used for each dataset in our experiments

Parameter	**EURLex-4K**	**LF-AmazonTitles-131K**	**LF-Wikipedia-500K**	**LF-AmazonTitles-1.3M**
Number of GPU	2	4	8	16
Total Training time (hrs)	0.27	4.2	37	66

B.7 DISTRIBUTED log SOFTTOP-k IMPLEMENTATION

Below we provide the code snippet for efficient distributed implementation of log SoftTop-k operator (for loss computation it is more desirable to directly work in log domain) in `PyTorch`, here it is assumed that `xs` (the input to our operator is of shape $B \times L/G$ where B is the batch size, L is the total number of labels and G is the number of GPUs used in the distributed setup.

```
# Topk code adapted from https://gist.github.com/thomasahle/4c1e85e5842d01b007a8d10f5fed3a18
sigmoid = torch.sigmoid
def sigmoid_grad(x):
    sig_x = sigmoid(x)
    return sig_x * (1 - sig_x)

from torch.autograd import Function
class DistLogTopK(Function):
    @staticmethod
    def forward(ctx, xs, k, alpha, n_iter=32):
        logits, log_sig = _dist_find_ts(xs, k, alpha=alpha, n_iter=n_iter, return_log_sig=True)
        ctx.save_for_backward(torch.tensor(alpha), logits)
        return log_sig

    @staticmethod
    def backward(ctx, grad_output):
        # Compute vjp, that is grad_output.T @ J.
        alpha, logits = ctx.saved_tensors
        sig_x = sigmoid(logits)
        v = alpha*sig_x*(1-sig_x)
        s = v.sum(dim=1, keepdims=True)
        dist.all_reduce(s, op=dist.ReduceOp.SUM)
        uv = grad_output * alpha * (1-sig_x)
        uv_sum = uv.sum(dim=1, keepdims=True)
        dist.all_reduce(uv_sum, op=dist.ReduceOp.SUM)
        t1 = - uv_sum * v / s
        return t1.add_(uv), None, None, None  # in-place addition

@torch.no_grad()
def _dist_find_ts(xs, k, alpha=1, n_iter=64, return_log_sig=False):
    assert alpha > 0
```

```
    b, n = xs.shape
    n *= dist.get_world_size()
    if isinstance(k, int):
        assert 0 < k < n
    elif isinstance(k, torch.LongTensor):
        assert (0 < k).all() and (k < n).all()
    # Lo should be small enough that all sigmoids are in the 0 area.
    # Similarly Hi is large enough that all are in their 1 area.
    xs_min = xs.min(dim=1, keepdims=True).values
    xs_max = xs.max(dim=1, keepdims=True).values
    dist.all_reduce(xs_min, op=dist.ReduceOp.MIN)
    dist.all_reduce(xs_max, op=dist.ReduceOp.MAX)
    lo = -xs_max - 10/alpha
    hi = -xs_min + 10/alpha
    for _ in range(n_iter):
        mid = (hi + lo)/2
        sigmoid_sum = sigmoid(alpha*(xs + mid)).sum(dim=1)
        dist.all_reduce(sigmoid_sum, op=dist.ReduceOp.SUM)
        mask = sigmoid_sum < k
        lo[mask] = mid[mask]
        hi[~mask] = mid[~mask]

    ts = (lo + hi)/2
    logits = alpha*(xs + ts)
    if return_log_sig:
        log_sig = logits - torch.logaddexp(logits, torch.zeros_like(
logits[:, :1]))
        return logits, log_sig
    else:
        return logits, sigmoid(logits)
```

C Additional Results and Analysis

C.1 Memorization capabilities of DE

In order to verify if DE models are capable of memorizing random text co-relations even in the presence of an extreme semantic gap between query and label text, we create synthetic datasets of different sizes with completely random query and label texts, and then train and evaluate retrieval performance of DE models. More specifically, for a synthetic dataset of size N, we sample N random query texts (by independently sampling 16 random tokens to form one query text) and similarly sample N random label texts. We assign i^{th} query to i^{th} label, train the DE model with in-batch and sampled hard negatives on this dataset, and then report the retrieval performance on the same dataset. As seen in Table 10, even at a million scale DE models perform this task with perfect accuracy which shows that they are well capable of performing random text memorization.

Table 10: Performance of DE on random text pair synthetic dataset at different scales

Dataset size (N)	**P@1**	**P@5**	**R@10**	**R@100**	**P@1**	**P@5**	**R@10**	**R@100**
	DE with in-batch negative				DE with hard negative			
100K	100.00	20.00	100.00	100.00	100.00	20.00	100.00	100.00
500K	98.60	20.00	100.00	100.00	100.00	20.00	100.00	100.00
1M	81.21	18.67	95.71	99.15	99.93	20.00	99.99	99.99

C.2 DecoupledSoftmax VS STANDARD SOFTMAX

C.2.1 ANALYTICAL GRADIENT ANALYSIS

Softmax gradient analysis:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\ell_i$$

$$\ell_i = \sum_{j\in P_i}\ell_{ij}, \text{where } P_i \text{ is the set of positive documents of label } i$$

$$\ell_{ij} = \log(\sigma_{ij}), \text{here } \sigma_{ij} = \frac{e^{s(q_i,d_j)}}{\sum_k e^{s(q_i,d_k)}}, \text{where } s(q,d) = x_q^T z_d$$

$$\implies \frac{\partial \ell_{ij}}{\partial z_{d_l}} = \begin{cases}(1-\sigma_{il})x_{q_i} & \text{if } l = j\\ -\sigma_{il}x_{q_i} & \text{otherwise}\end{cases}$$

$$\implies \frac{\partial \ell_{i}}{\partial z_{d_l}} = \begin{cases}(1-|P_i|\sigma_{il})x_{q_i} & \text{if } l \in P_i\\ -|P_i|\sigma_{il}x_{q_i} & \text{otherwise}\end{cases}$$

This suggests that the multi-label extension of standard softmax is optimized when:

$$\sigma_{il} = \begin{cases}\frac{1}{|P_i|} & \text{if } l \in P_i\\ 0 & \text{otherwise}\end{cases}$$

DecoupledSoftmax **gradient analysis**:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\ell_i$$

$$\ell_i = \sum_{j\in P_i}\ell_{ij}, \text{where } P_i \text{ is the set of positive documents of label } i$$

$$\ell_{ij} = \log(\sigma_{ij}), \text{here } \sigma_{ij} = \frac{e^{s(q_i,d_j)}}{e^{s(q_i,d_j)} + \sum_{k\notin P_i} e^{s(q_i,d_k)}}, \text{where } s(q,d) = x_q^T z_d$$

$$\frac{\partial \ell_{ij}}{\partial z_{d_l}} = \begin{cases}(1-\sigma_{il})x_{q_i} & \text{if } l = j\\ 0 & \text{if } l \neq j \text{ and } l \in P_i\\ -\frac{e^{s(q_i,d_l)}}{e^{s(q_i,d_j)}+\sum_{k\notin P_i} e^{s(q_i,d_k)}}x_{q_i} & \text{otherwise}\end{cases}$$

$$\implies \frac{\partial \ell_{i}}{\partial z_{d_l}} = \begin{cases}(1-\sigma_{il})x_{q_i} & \text{if } l \in P_i\\ -\sum_{j\in P_i}\frac{e^{s(q_i,d_l)}}{e^{s(q_i,d_j)}+\sum_{k\notin P_i} e^{s(q_i,d_k)}}x_{q_i} & \text{otherwise}\end{cases}$$

C.2.2 EMPIRICAL RESULTS ON XMC BENCHMARKS

Table 11: Side by side comparison of DE models trained with DecoupledSoftmax vs Softmax loss on XMC datasets.

Dataset	**P@1**		**P@5**	
	Decoupled Softmax	Softmax	Decoupled Softmax	Softmax
EURLex-4K	86.78	80.05	60.19	58.36
LF-AmazonTitles-131K	42.52	41.77	20.64	20.87
LF-Wikipedia-500K	85.78	79.85	50.53	49.78
LF-AmazonTitles-1.3M	58.40	52.42	45.46	43.43

C.2.3 COMPARING DecoupledSoftmax AND SOFTMAX WITH HARD NEGATIVES

Table 12: Performance comparison of DecoupledSoftmax vs Softmax with sampled hard negatives, note that Softmax with sampled hard negatives is same as InfoNCE with hard negatives

Batch size	Hard neg per query	Effective doc pool	P@1	P@5	R@10	R@100	P@1	P@5	R@10	R@100
			DecoupledSoftmax				Softmax			
8192	1	~ 16384	49.16	39.07	32.76	60.09	47.60	38.85	32.12	61.49
8192	2	~ 24576	50.74	40.14	33.31	60.45	49.11	40.16	32.77	62.05
8192	10	~ 90112	54.01	42.08	34.19	61.04	51.19	42.06	33.31	62.76
8192	All	$\sim 1.3M$	58.40	45.46	36.49	64.25	52.42	43.42	34.00	65.84

C.3 QUALITY OF NEGATIVES IN DE TRAINING

As mentioned in Section 5.4 we perform negative sampling following (Xiong et al., 2021), we mine hard negatives by computing the shortlist of nearest indexed labels for each training query at regular intervals and sample negatives from this shortlist. Although, this approach is computationally efficient it suffers from the problem of having to deal with stale negatives since the model parameters constantly change but the negative shortlist gets updated only after certain training steps. To study the importance of the quality of hard negatives we implement an approach similar to (Lindgren et al., 2021) where we maintain an active cache of all label embeddings and use this to first identify the hardest negatives and then use these negatives along with in-batch labels to perform loss computation. More specifically, we maintain a label embedding cache of $L \times d$ size where L is the number of labels and d is the embedding dimension. We initialize this matrix with the embeddings of all the labels at the start of training. For a mini-batch of B queries and L' sampled labels in the mini-batch, we first update the embedding cache of all L' mini-batch labels, then we compute the loss using embeddings of B training queries and the label embedding cache and identify the labels that receive the highest gradients. We take the top-K labels that are not already part of the L' mini-batch labels. Then we use these top-k labels as the hard negative for the mini-batch and then compute the loss using the L' mini-batch labels and K hard negative labels. This approach allows us to mine *almost* fresh hard negatives for each mini-batch. In Table 13 we compare the results obtained using this approach and ANCE (Xiong et al., 2021) style hard negatives. Note that for a fair comparison, we compare them in settings where an equivalent total number of negatives are sampled per batch. These results suggest that there can be moderate gains made with an increase in the quality of negatives and further research into mining better negatives can be helpful in training better DE models with lower cost.

Table 13: Comparing DecoupledSoftmax DE trained with ANCE (Xiong et al., 2021) style stale negatives and fresh negatives mined using embedding cache (Lindgren et al., 2021)

Dataset	P@1	P@5	R@10	R@100	P@1	P@5	R@10	R@100
	ANCE (Xiong et al., 2021)				Embedding cache (Lindgren et al., 2021)			
EURLex-4K	84.61	59.15	71.08	90.21	85.87	59.93	71.54	90.34
LF-AmazonTitles-131K	40.99	20.59	57.20	70.08	41.72	20.25	45.54	65.54
LF-Wikipedia-500K	82.85	48.84	74.47	90.08	84.53	48.66	72.97	88.93

C.4 RESULTS WITH CONFIDENCE INTERVALS ON SMALL DATASETS

Since XMC datasets are fairly big benchmarks so running multiple experiments with different seeds is very expensive. Below in Table 14 we provide mean and 95% confidence intervals for DecoupledSoftmax DE results for the smaller EURLex-4K and LF-AmazonTitles-131K datasets.

C.5 SCORE DISTRIBUTION ANALYSIS

Table 14: Results with 95% confidence intervals for DecoupledSoftmax DE trained on EURLex-4K and LF-AmazonTitles-131K dataset

P@1	P@5	R@10	R@100	P@1	P@5	R@10	R@100
EURLex-4K				LF-AmazonTitles-131K			
86.50 ± 0.24	60.24 ± 0.19	72.62 ± 0.26	91.76 ± 0.10	42.37 ± 0.13	20.57 ± 0.05	56.25 ± 0.09	68.51 ± 0.04

In Figure 5 we plot the precision-recall curve on EURLex-4K for DE models trained using DecoupledSoftmax and Softmax loss. This plot reveals that DE model trained with DecoupledSoftmax offers a better precision vs recall tradeoff. In Figure 6 we analyze the distribution of scores given by DE models trained with different loss function on the test set of EURLex-4K dataset. More specifically, we plot the distribution of scores of positive and negative labels for Softmax loss 3, DecoupledSoftmax loss 5, and SoftTop-5 loss 6. These plots reveal that the positives and negatives are better separated when we use DecoupledSoftmax and SoftTop-5 loss, when compared to the Softmax loss.

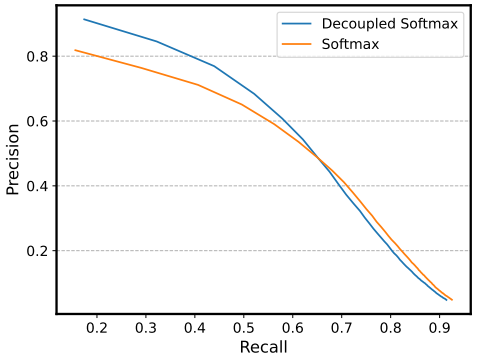

Figure 5: PR curve for Decoupled softmax vs standard softmax on EURLex-4K dataset

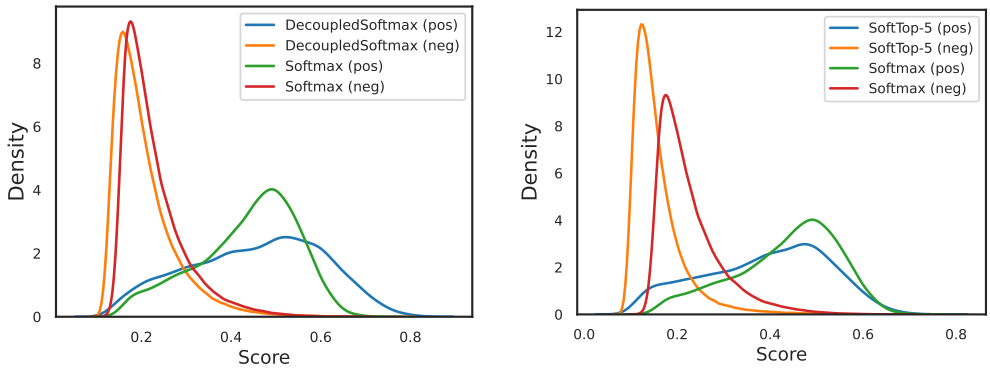

Figure 6: Plot of score distribution of DE models trained with different loss functions. ***left*** plot compares DecoupledSoftmax 5 vs softmax 3 ***right*** plot compares SoftTop-5 6 vs softmax 3

C.6 DECILEWISE COMPARISON

We investigate the performance variation across different label regimes (many-shot versus few-shot labels) among various approaches. These include the pure classification-based method (DistilBERT OvA Classifier), DecoupledSoftmax DE models trained with in-batch negatives (In-batch DE), hard negatives (Hard negative DE), and all negatives (DecoupledSoftmax DE and SoftTop-k DE). Consistent with Dahiya et al. (2023a), we devise different label deciles based on the count of training examples per label, wherein the first decile signifies labels with the most training examples and the last decile signifies those with the least. On LF-Wikipedia-500K, as anticipated, Figure 7 shows that the pure classification-based approach achieves subpar performance on the last deciles. However, intriguingly, DE methods trained with hard or all negatives exhibit performance comparable to the classification-based approaches even on head deciles. DE methods trained with our approach does well across *all* deciles and hence improve the performance over existing DE methods. On LF-AmazonTitles-1.3M, when comparing the DistilBERT OvA classifier with DE models, we notice a similar trend that the pure classification-based approach and DE approach are comparable at head deciles but on tail deciles DE starts significantly outperforming the classifier approach. When comparing DE trained with all negatives and the ones trained with in-batch or hard-negatives, we notice a substantial gap on head deciles but the gap diminishes on tail deciles.

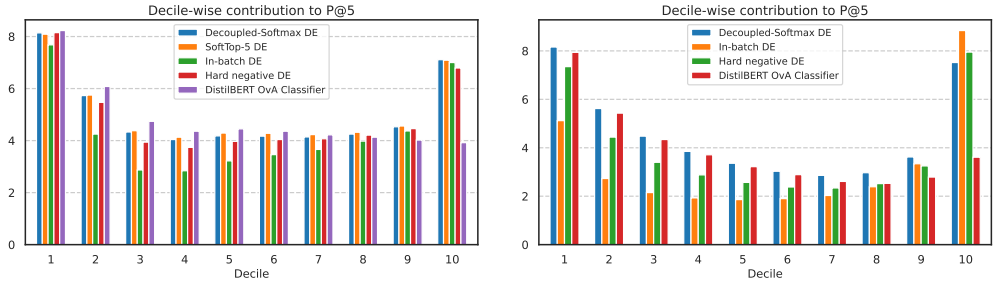

Figure 7: Decile-wise analysis of P@5 on LF-Wikipedia-500K (***left***) and LF-AmazonTitles-1.3M (***right***), 1^{st} decile represents labels with most training points i.e. head labels while 10^{th} decile represents labels with least training points i.e. tail labels

C.7 Better generalization of DE models on tail labels

In this section, we study whether DE models offer improved learning when dealing with similar but distinct labels when compared to the classifier approach. We use LF-Wikipedia-500K dataset for this experiment. We first quantify how much a label is similar to the rest of the labels in the dataset, we do this by computing the normalized embeddings of each label (from the trained DE model) and searching for the top 10 nearest labels for each label, we use the mean similarity score of the top 10 nearest labels as an indication of how similar this label is compared to rest of the labels. Based on this mean similarity score, we divide our labels into 3 equal bins, where the first bin corresponds to the *most dissimilar* labels and the last bin corresponds to the *most similar* labels. We compute the contribution of labels from each bin to final P@5 numbers for both the pure classifier-based approach (DistilBERTOvA baseline) and pure dual encoder-based approach (DecoupledSoftmax DE). We then plot the relative improvement between the classifier-based approach and the dual-encoder approach, more specifically we plot the relative improvement in P@5 contribution of the dual-encoder method relative to the classifier method. We also perform the same analysis but this time we create label bins for only tail labels (we define a tail label as any label with ≤ 5 training points

From Figure 8 we can infer that there doesn't seem to be any meaningful pattern when we look at the results of the analysis on all labels but when we look at the analysis just on tail labels there appears to be a trend that the relative improvement is more on similar labels and less on dissimilar labels. This demonstrates that for labels with less training data, dual-encoders can perform relatively better on similar but distinct labels compared to a pure classifier-based approach which treats each label as an atomic entity.

C.8 Comparison with XR-Transformer on EURLex-4K

As highlighted in (Gupta et al., 2022), methods like XR-Transformer use high-capacity sparse classifiers learned on the concatenated sparse tf-idf features and dense embedding obtained from the BERT encoder for ranking their top predictions. In table 15 we compare XR-Transformer's performance without ensembling and without using tf-idf features. We also, report results by adding the sparse ranker layer (implemented similarly to the approach described in Section 3.5 of (Gupta et al., 2022)) on top of DecoupledSoftmax DE top 100 predictions. As we can see that XR-Transformer's better performance on EURLex-4K dataset can be greatly attributed to the use of ensemble of 3 learners and high-capacity sparse classifiers.

Table 15: Comparison with XR-Transformer on EURLex-4K dataset

Loss	**P@1**	**P@3**	**P@5**
XR-Transformer $\times 3$ ensemble	88.41	75.97	63.18
XR-Transformer	87.19	73.99	61.60
XR-Transformer w/o tf-idf ranker	83.93	69.48	56.62
DecoupledSoftmax DE	86.78	73.40	60.19
DecoupledSoftmax DE with tf-idf ranker	86.55	75.21	62.26

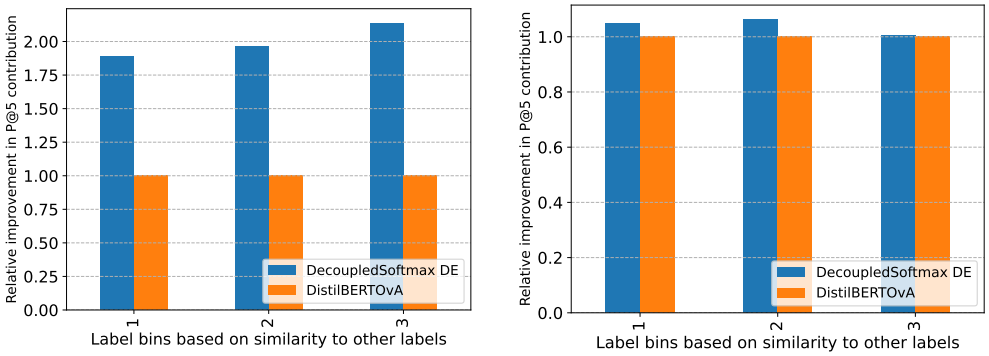

Figure 8: Plot to study if dual encoder training performs better on similar but distinct labels on LF-Wikipedia-500K dataset. We first divide labels into 3 equal bins, where the first bin corresponds to the *most dissimilar* labels and the last bin corresponds to the *most similar* labels (based on the mean similarity score of the top 10 nearest embedded labels). We then compute the contribution of labels from each bin to final P@5 numbers for both pure classifier-based approach (DistilBERTOvA baseline) and pure dual encoder-based approach (DecoupledSoftmax DE). In the **right** subplot, we plot the relative improvement between the classifier based approach and the dual encoder approach when considering all labels, more specifically we plot the relative improvement in P@5 contribution of the dual encoder method relative to the classifier method. In the **left** perform the same analysis but this time we create label bins for only tail labels (we define a tail label as any label with ≤ 5 training points.