Machine Learning (2024) 113:675–697
https://doi.org/10.1007/s10994-023-06468-w

Meta-classifier free negative sampling for extreme multilabel classification

Mohammadreza Qaraei[1] · **Rohit Babbar**[1,2]

Received: 14 February 2023 / Revised: 15 June 2023 / Accepted: 17 October 2023 /
Published online: 20 November 2023

Abstract

Negative sampling is a common approach for making the training of deep models in classification problems with very large output spaces, known as extreme multilabel classification (XMC) problems, tractable. Negative sampling methods aim to find per instance negative labels with higher scores, known as hard negatives, and limit the computations of the negative part of the loss to these labels. Two well-known methods for negative sampling in XMC models are meta-classifier-based and Maximum Inner product Search (MIPS)-based adaptive methods. Owing to their good prediction performance, methods which employ a meta classifier are more common in contemporary XMC research. On the flip side, they need to train and store the meta classifier (apart from the extreme classifier), which can involve millions of additional parameters. In this paper, we focus on the MIPS-based methods for negative sampling. We highlight two issues which may prevent deep models trained by these methods to undergo stable training. First, we argue that using hard negatives excessively from the beginning of training leads to unstable gradient. Second, we show that when all the negative labels in a MIPS-based method are restricted to only those determined by MIPS, training is sensitive to the length of intervals for pre-processing the weights in the MIPS method. To mitigate the aforementioned issues, we propose to limit the labels selected by MIPS to only a few and sample the rest of the needed labels from a uniform distribution. We show that our proposed MIPS-based negative sampling can reach the performance of LightXML, a transformer-based model trained by a meta classifier, while there is no need to train and store any additional classifier. The code for our experiments is available at https://github.com/xmc-aalto/mips-negative-sampling.

Keywords Extreme classification · Negative sampling · Hard negative mining · Maximum inner product search · Deep neural networks

Editors: Fabio Vitale, Tania Cerquitelli, Marcello Restelli, Charalampos Tsourakakis.

✉ Mohammadreza Qaraei
mohammadreza.mohammadniaqaraei@aalto.fi

[1] Aalto University, Helsinki, Finland

[2] University of Bath, Bath, UK

1 Introduction

Machine learning problems with large output spaces are common in a variety of domains such as face recognition (Schroff et al., 2015), open-domain question answering (Lee et al., 2019), and language models (Bengio & Senécal, 2008). For multilabel classification problems, consisting of hundreds of thousand or even millions of labels, this is referred to as Extreme Multilabel Classification (XMC) (Bhatia et al., 2016). Beyond automatic tagging of web-scale corpora, such as Wikipedia (Partalas et al., 2015), the framework of XMC can be leveraged to address search, recommendation and web-advertising (Dahiya et al., 2021a; Jain et al., 2019).

The main challenge in XMC is the enormous computation needed for training a model due to the large label space. In early approaches for XMC, to partially mitigate this issue, the training of binary linear classifiers was done in parallel by exploiting the fact that under a one-vs-rest setting, learning a binary classifier for every label is independent of the rest of the labels (Babbar & Schölkopf, 2017, 2019; Schultheis & Babbar, 2022). Furthermore, several methods employed label trees to achieve computational complexity which is logarithmic in the number of labels (Khandagale et al., 2020; Prabhu et al., 2018).

For training using deep neural networks, computing the loss and its gradient over a mini-batch requires a pass over all the labels meaning that the computations grow linearly with the number of labels at each iteration of the training algorithm. A common approach to make training feasible in deep XMC models is to employ negative sampling methods, discussed next, to approximate the loss and its gradient.

1.1 Negative sampling in XMC

Negative sampling methods use the fact that there are only a few positive labels for each training point, while the rest of them, also called the negative labels, is extremely large. Therefore, if only a small fraction of negative labels along with all the positive labels are involved in computing the loss, the computational complexity of training the model will decrease significantly. There are three main approaches for negative sampling: static, meta-classifier-based, and adaptive methods.

In *static* negative sampling, the negative labels are drawn from a fixed distribution independent of the model (Mikolov et al., 2013). While static methods are fast, their approximation of the full loss is known to be biased (Blanc & Rendle, 2018; Rawat et al., 2021), and hence the resulting models perform poorly unless the number of sampled labels is large.

Meta-classifier-based methods have recently become popular for training deep XMC models. These methods train an additional classifier, known as meta classifier, in addition to the extreme classifier, which is responsible for suggesting confusing negative labels to the extreme classifier (Dahiya et al., 2021b; Jiang et al., 2021; Kharbanda et al., 2021, 2022; You et al., 2019; Zhang et al., 2021).

While meta-classifier-based methods can lead to high accuracy, this is achieved at the cost of training and storing a meta classifier which can be as large as the extreme classifier. Also, no matter how accurate the meta classifier is, the suggested negative labels may not be the most confusing ones for the extreme classifier since the meta classifier is usually fully (You et al., 2019; Zhang et al., 2021) or partially (Jiang et al., 2021; Kharbanda et al., 2021) independent of the extreme classifier.

Adaptive negative sampling methods leverage the fact that sampling from a distribution proportional to the scores of the labels leads to an unbiased estimate of the full loss (Blanc & Rendle, 2018; Rawat et al., 2019). In models where the scores of the labels are computed by an inner product operation such as neural networks, a close approach to this is to use a Maximum Inner Product Search (MIPS) for finding the labels with the highest scores and restricting the negative part of the loss to these labels.

However, an exact MIPS needs an exhaustive search over all the labels, which is expensive. An alternative way is to employ approximate MIPS methods, which are initially designed to accelerate inference in machine learning models. They involve building an index over the weights of the classifier as a pre-processing step, and during inference, the index is utilized to approximately identify the labels with high scores (Auvolat et al., 2015; Johnson et al., 2019; Shrivastava & Li 2014).

For training, on the other hand, applying approximate MIPS for negative sampling during training classifiers is not straightforward due to its high computational complexity. This is mainly because, unlike inference where a fixed index is used, the weights of the classifier are changing constantly during training so the index built by the MIPS module needs to be updated frequently. To mitigate this issue, a common practice is to perform pre-processing only a few times during training. Furthermore, two studies aimed to speed up approximate MIPS methods during query time when employed for negative sampling (Daghaghi et al., 2021; Yen et al., 2018). In Daghaghi et al. (2021), this is achieved by utilizing local sensitivity hashing (LSH) as a sampling mechanism instead of the top-k search, and Yen et al. (2018) proposed to replace the high dimensional MIPS with several lower dimensional ones.

In contrast to meta-classifier-based methods, adaptive negative sampling methods do not need to train and store any extra model. However, to the best of our knowledge, the existing adaptive methods are inferior to meta-classifier-based methods in terms of prediction accuracy and their performance in deep neural networks, like transformer-based models, has remained under-explored.

1.2 Meta classifier free negative sampling

In this paper, we highlight difficulties in training deep XMC models using MIPS-based adaptive negative sampling methods when the negative part of the loss is restricted to only the labels suggested by the MIPS module. We hypothesise two reasons for this.

First, when negative labels are restricted to the hardest ones from the beginning of training, the gradient with respect to the embedding vector is dominated by the term related to the negative labels, which prevents the embeddings from becoming similar enough to the weights corresponding to the positive labels. To mitigate this issue, one can incorporate a warm-up phase into training by using only random negative labels for a few iterations and switching to the labels found by the MIPS module for the rest of the training (Yen et al., 2018).

Second, we show that MIPS-based adaptive negative sampling methods are highly sensitive to the length of the intervals between the successive (weights) pre-processing steps. If these intervals are large, the weights related to a few negative labels frequently suggested by the MIPS module will be penalized over and over. This turns those negative labels from hard negatives to easy negatives, while these continue to be among the labels suggested by the MIPS module until the next pre-processing step.

To avoid the aforementioned issues in training deep models using MIPS-based methods, in this paper, we propose to combine top candidate labels found by MIPS with several labels drawn from a static distribution. Specifically, to build the set of negative labels for an instance, we pick approximately *#positives* labels through the MIPS module and select the rest ($\approx 0.01 \times L$) from a uniform distribution, where *#positives* indicates the average number of positive labels per sample and L is the number of labels in our dataset. Such a combination not only stabilizes the gradient in the initial steps of the training but also mitigates the problem of having only easy negatives even when the intervals for pre-processing the weights in the MIPS module are less frequent. In the experimental section, we compare the proposed method with a MIPS-based negative sampling method in which all the negative labels used in approximating the loss are those predicted by the MIPS module. The results on three extreme text classification datasets show that the proposed method achieves significantly higher precision, particularly in the presence of large pre-processing intervals.

Finally, we apply the proposed method to a transformer-based model, demonstrating that the architecture of this model, which utilizes a clustering-based MIPS module, is very similar to LightXML (Jiang et al., 2021), a negative sampling approach based on meta classifiers. Specifically, in our method, the MIPS module can be seen as the counterpart to LightXML's meta classifier. However, since the MIPS module works directly on the weights of the extreme classifier, it does not require extra training and does not add additional parameters to the model.

The results on two extreme classification datasets show that the proposed negative sampling method is similar to LightXML in terms of prediction performance, while being up to 33% smaller in terms of model size. This suggests that if the negative labels are engineered well in adaptive negative sampling methods, these methods can reach meta-classifier-based ones in terms of prediction performance, while they have smaller model sizes and can lead to lower training time as there is no need to train and store any meta classifier.

1.3 Contributions

To summarize, our key contributions are as follows:

- We highlight difficulties in training deep models using MIPS-based negative sampling methods. We show that when the negative labels are restricted to only hard negatives from the beginning of training the deep model fails to train properly. Also, when the intervals for pre-processing the weights are large, the labels used for approximating the loss may not consist informative negative labels.
- We propose to pick only a few labels from the MIPS and select the rest of the labels from a uniform distribution, which mitigates the aforementioned issues in training deep models using MIPS-based negative sampling.
- We compare the proposed method with LightXML, a meta-classifier-based negative sampling approach, on a very similar architecture. The results show that our proposed MIPS-based negative sampling can achieve accuracy similar to LightXML, while there is no need to train and store a relatively large meta classifier.

2 Negative sampling

Assume a training set $\{\mathrm{x}_i, \mathrm{y}_i\}_{i=1}^N$ is given, where $\mathrm{x}_i \in \mathbb{R}^D$ are the features of the i-th sample and $\mathrm{y}_i \in \{1, 0\}^L$ are the labels. We denote the embedded features for sample x by $E_\mathrm{x} \in \mathbb{R}^d$. In extreme multilabel classification, the goal is to learn a score function $f : \mathbb{R}^d \rightarrow \mathbb{R}^L$ over embeddings E_{x_i} such that for a pair (x, y), $f_j(E_\mathrm{x})$ has a high value when $y_j = 1$. The score function $f(.)$ is usually a linear function, which can be formulated as follows:

$$f(E_\mathrm{x}) = W^T E_\mathrm{x} \tag{1}$$

where $W \in \mathbb{R}^{d \times L}$ are the parameters of the score function that are learned during training and E_x is produced by an encoder, which can be for instance a deep neural network.

In XMC, most of the loss functions are the sum of binary losses such as BCE or hinge loss, and are computed as follows:

$$l(f(E_\mathrm{x}), \mathrm{y}) = \sum_{j:y_j=1} l_+(f_j(E_\mathrm{x})) + \sum_{j:y_j=0} l_-(f_j(E_\mathrm{x})) \tag{2}$$

where $l_+, l_- : \mathbb{R} \rightarrow \mathbb{R}_+$ are the positive and negative parts of the binary loss.

In extreme classification problems, computing the negative part of the loss in Eq. (2) becomes intractable as the set $\mathcal{N} = \{j | y_j = 0\}$ is extremely large. To overcome this problem, negative sampling methods compute the negative part of the loss over a subset of all possible negative labels, $\hat{\mathcal{N}} \subset \mathcal{N}$, where $|\hat{\mathcal{N}}| \ll |\mathcal{N}| \approx L$. This leads to the following formulation for the loss function:

$$\hat{l}(f(E_\mathrm{x}), \mathrm{y}) = \sum_{j:y_j=1} l_+(f_j(E_\mathrm{x})) + \sum_{j \in \hat{\mathcal{N}}} l_-(f_j(E_\mathrm{x})) \tag{3}$$

Negative sampling methods are different in terms of the choice of $\hat{\mathcal{N}}$. In order to minimize $|l - \hat{l}|$ as much as possible, several works suggested choosing $\hat{\mathcal{N}}$ in a way that it contains hard negatives meaning that $f_j(E_\mathrm{x})$ is higher than a specific threshold for all $j \in \hat{\mathcal{N}}$. One way to efficiently find these hard negatives during training is to use an approximate Maximum Inner Product Search (MIPS). However, this may lead to difficulties in training deep models, which are discussed in the next section.

3 Difficulties in training by MIPS-based negative sampling methods

In this section we hypothesise two reasons which can lead to poor results in ordinary MIPS-based negative sampling methods. All the experiments conducted in this section are on Eurlex, a document classification dataset with ≈ 4000 labels (Bhatia et al., 2016), using either a linear classifier or a neural network with one hidden layer.

3.1 Exact hard negative mining from scratch

Figure 1 illustrates a comparison of a linear model and a neural network when the negatives are restricted to be the top-100 labels returned by an exact MIPS method. The results show

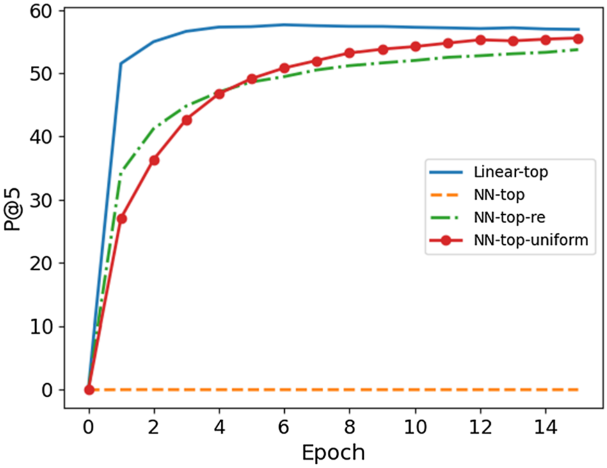


Fig. 1 A comparison of precision at 5 (P@5) in a linear classifier and a neural network trained using hard negatives on the Eurlex dataset. The figure shows the linear classifier can be trained properly using the hardest negatives, while the output of the neural network (NN-top) is just a random guess. Using higher learning rates for the positive part of the loss (NN-top-re) or picking a few hard negatives and combining them with randomly sampled labels (NN-top-uniform) can fix the issue

that, with these as the negatives comprising the set $\hat{\mathcal{N}}$, the linear model (Linear-top) can be trained properly, while the output of the neural network (NN-top) is not better than a random guess.

We hypothesise that the reason for stable training in linear models is the independence of the parameters learned for each label. Specifically, the only learnable parameters in the linear model are the weights of the classifier, and no matter what other negative labels are included in computing the loss, the gradient will be the same for the weights connected to a particular label. However, when the embeddings are learned during training, such as in neural networks, the gradient with respect to an embedding will be affected by the negative labels used in computing the loss. In this case, the negative sampling method has a big impact on learning the embeddings.

More specifically, without loss of generality, assume the loss in Eq. 2 is the BCE loss:

$$l_+(f_i(E_\mathrm{x})) = -\log(\sigma(f_i(E_\mathrm{x}))),$$
$$l_-(f_i(E_\mathrm{x})) = -\log(1-\sigma(f_i(E_\mathrm{x})))$$

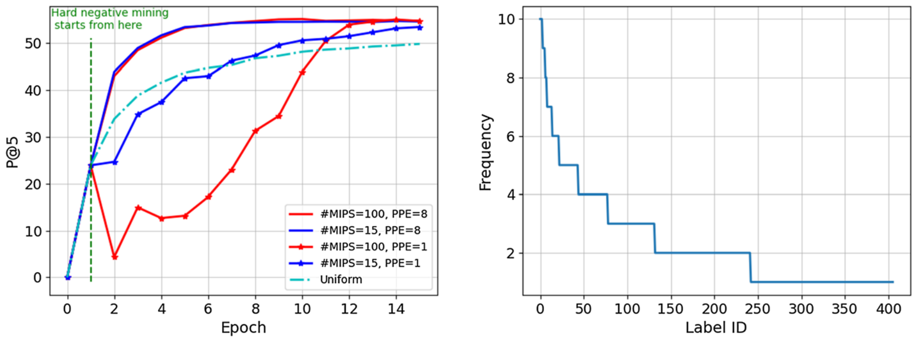
Then, the gradient of the loss with respect to E_x is as follows:

$$\nabla_{E_\mathrm{x}} l = \sum_{j:y_j=1}(o_j-1)\mathrm{w}_j + \sum_{j\in\mathcal{N}} o_j\mathrm{w}_j \tag{4}$$

where $o_j := \sigma(f_j(E_\mathrm{x}))$ and w_j is the parameter vector of the extreme classifier corresponding to label j. Then a single step of the stochastic gradient descent algorithm for updating the embedding will be as follows:

$$E_\mathrm{x} \leftarrow E_\mathrm{x} + \eta^+ \sum_{j:y_j=1}(1-o_j)\mathrm{w}_j - \eta^- \sum_{j\in\mathcal{N}} o_j\mathrm{w}_j \tag{5}$$

where η^+ and η^- are the learning rates and typically $\eta^+ = \eta^-$. This equation suggests that the optimal embedding for a sample should be similar to the weights of the extreme classifier for the positive labels and be dissimilar to those for the negative labels. However, if the summation over the negative weights is restricted to only a few hardest negatives (those with the highest o_j), then this term will have a high magnitude and the total gradient will be



(a)

(b)

Fig. 2 **a** A comparison of P@5 for different numbers of negative labels selected by MIPS (#MIPS) as well as different numbers of checkpoints for pre-processing the weights per epoch (PPE). For the case that all the selected negative labels are determined by MIPS and pre-processing intervals are large (#MIPS=100, PPE=1), P@5 has an unstable behaviour, while restricting the negative labels from MIPS to 15 and sampling the rest from a uniform distribution (#MIPS=15, PPE=1) mitigate the problem. **b** Distribution of the hardest negative labels after switching to hard negative mining from uniform negative sampling

only determined by this term. This will prevent the embedding to become close enough to the weights corresponding to the positive labels while it may also not become dissimilar to the other negative weights which are not included in computing the loss. To mitigate this problem, one can force the optimizer to take a larger step towards the positive weights by choosing $\eta^+ > \eta^-$. Figure 1 shows that in case $\frac{\eta^+}{\eta^-} = 50$, the model will be able to train properly when all the negative labels are the hardest ones. However, the optimal ratio of η^+ and η^- will add a hyperparameter to the model.

Another possible way to stabilize training when hard negatives are used, which has been employed by Xiong et al. (2020) and Yenet al. (2018), is to do negative sampling from a static distribution for a few iterations and then switch to using hard negatives for the rest of training. The initial steps of negative sampling from a static distribution allow the embeddings to become similar to the positive weights of the extreme classifier.

In this paper, to have the benefits of using hard negatives while avoiding an unstable gradient, we propose to combine only a few hard negatives with several negative labels sampled from a static distribution. Using randomly sampled negative labels mitigates the problem of taking large steps towards the direction that is suggested by taking the hard negatives.

3.2 Impact of the length of pre-processing intervals

Most of the adaptive negative sampling methods using MIPS perform pre-processing of the weights within specific intervals during training since the ideal case, which is pre-processing the weights at every iteration, is highly time-consuming. Assuming that we are using an exact MIPS method, this is equivalent to storing the weights of the classifier at specific checkpoints and using them for querying until the next checkpoint.

In this subsection, we investigate the effect of having relatively large intervals for pre-processing the weights on training a neural network. Figure 2a shows precision at 5 (P@5) for a neural network trained by 100 negative labels for approximating the loss when the pre-processing is done only once versus doing it 8 times per epoch. For mitigating issues related to using hard negatives from scratch mentioned in the previous subsection, we train the models only by uniform negative sampling for one initial epoch. The figure shows that for the case that there is only one pre-processing checkpoint per epoch, P@5 has an unstable behaviour especially in the few first epochs after switching to hard negative mining. However, P@5 looks more stable by restricting the negative labels suggested by MIPS to 15 and sampling the rest from a uniform distribution.

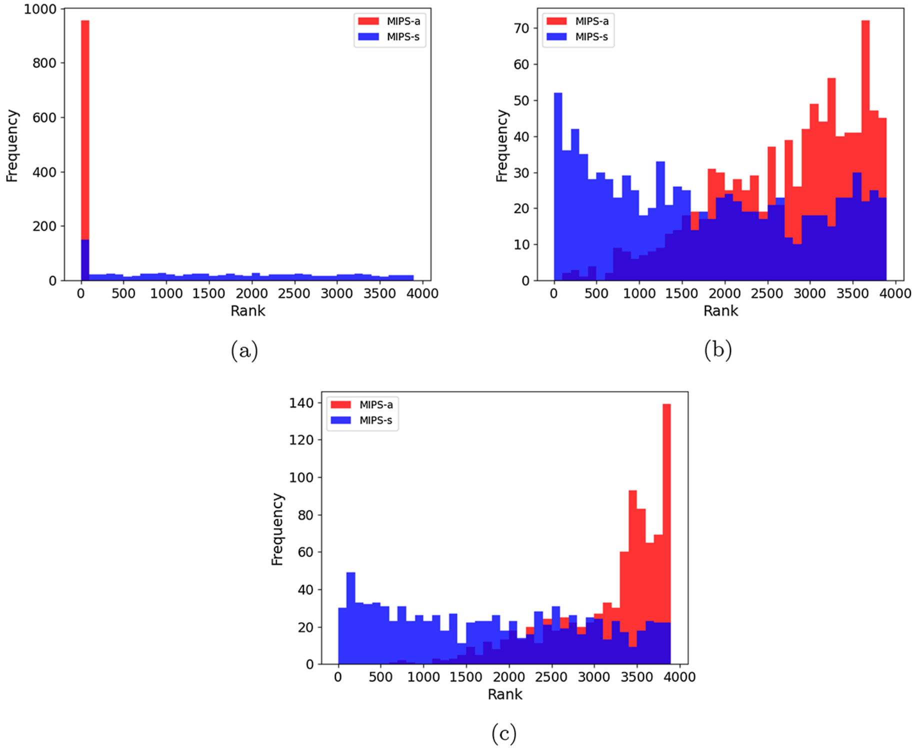
To answer the question of why the performance declines after switching to hard negative mining when the pre-processing intervals are large and all the negative label are selected through MIPS, we investigate the negative labels retrieved by the MIPS. Figure 2b shows that, after training the model with uniform negative sampling for one epoch, for a mini-batch of size 50 randomly drawn from the training set, the negative labels retrieved by the MIPS have an imbalanced distribution. Since the weights in the MIPS module are fixed until the next checkpoint for pre-processing, if the embeddings do not change much during training, most of the labels suggested by the MIPS module will be the same for all the training samples, and the weights corresponding to these labels will be penalized frequently within that epoch of training.

Figure 3 illustrates the true rank of the negative labels used for approximating the loss when only one pre-processing is done per epoch for the case that all the labels are selected by the MIPS method (MIPS-a) as well as the case that the labels from the MIPS method are restricted to 15 and the rest are sampled uniformly (MIPS-s). The histograms were computed three times after switching to using the MIPS method: right after the checkpoint for pre-processing the weights, in the middle, and also right before the next checkpoint for pre-processing. Figure 3c shows that for the case that all the negative labels are selected by the MIPS method, in the last iteration before the next checkpoint, contrary to what is expected, the labels retrieved by the MIPS method are only easy negatives and therefore do not contribute much in training the model. However, for the MIPS-s method, the negative labels approach a uniform distribution.

4 Proposed method

Our proposed method for performing hard negative mining while avoiding unstable training in deep models is to find and select a few labels with high scores and sample the rest of the needed labels from a static distribution. To find the labels with high scores in a reasonable time during training, we use a clustering-based approximate MIPS (Auvolat et al., 2015; Johnson et al., 2019). More precisely, as the pre-processing step, we cluster the weights of the extreme classifier using spherical K-means, which partitions the weights according to their directions, at specific checkpoints. Then, for each training sample, to approximate the labels with high scores at the query time, we search a few clusters whose centroids maximize the inner product with the embedded features obtained by the encoder for that sample. The details of the pre-processing and querying steps of the approximate MIPS used in our method are described in the following.

Pre-processing The pre-processing is done by clustering the weights of the extreme classifier using spherical K-means (Auvolat et al., 2015; Zhong, 2005). More precisely, we randomly select K unit-length vectors, $C = \{\mathrm{c}_1, ..., \mathrm{c}_K\}$, and repeat the following two steps until meeting the convergence criteria (Auvolat et al., 2015):



(a)

(b)

(c)

Fig. 3 Frequency of the true ranks (based on the model's predictions) of the negative labels used for approximating the loss for the case that all the 100 negative labels are selected by the MIPS method (MIPS-a) as well as restricting the labels from the MIPS method to 15 and sampling the rest from a uniform distribution. Histograms were computed in the first iteration after pre-processing the weights (**a**), in the middle iteration (**b**), and in the last iteration before the next checkpoint for pre-processing the weights (**c**). The histograms show that for the case that the distance from the pre-processing checkpoint is large, there are only labels with high ranks (easy negatives) among the labels of MIPS-a, while the negative labels of MIPS-s show a uniform distribution

1. For each weight vector w_j, we compute the index of the cluster to which this weight vector belongs by $a_j = \operatorname{argmax}_{i \in \{1,\dots,K\}} \mathrm{w}_j^T \mathrm{c}_i$.
2. For each cluster i, we estimate its centroid by $\mathrm{c}_i = \frac{\sum_{j|a_j=i} \mathrm{w}_j}{\| \sum_{j|a_j=i} \mathrm{w}_j \|}$

Since the pre-processing step is time-consuming, we perform it only at specific checkpoints during training.

Querying Let the weights of the extreme classifier at checkpoint h be $W^h_{d \times L}$, and $C^h = \{\mathrm{c}^h_1, \dots, \mathrm{c}^h_K\}$ be the set of centroids obtained by clustering these weights. Also, let I_j be the label ids of the weights inside the j-th cluster. As the first step of querying, a search over the centroids is done to find τ centroids which maximize the inner product with the encoded sample E_x:

$$C^* = \tau\text{-}\operatorname*{argmax}_{\mathrm{c} \in C^h} E_\mathrm{x}^T \mathrm{c} \tag{6}$$

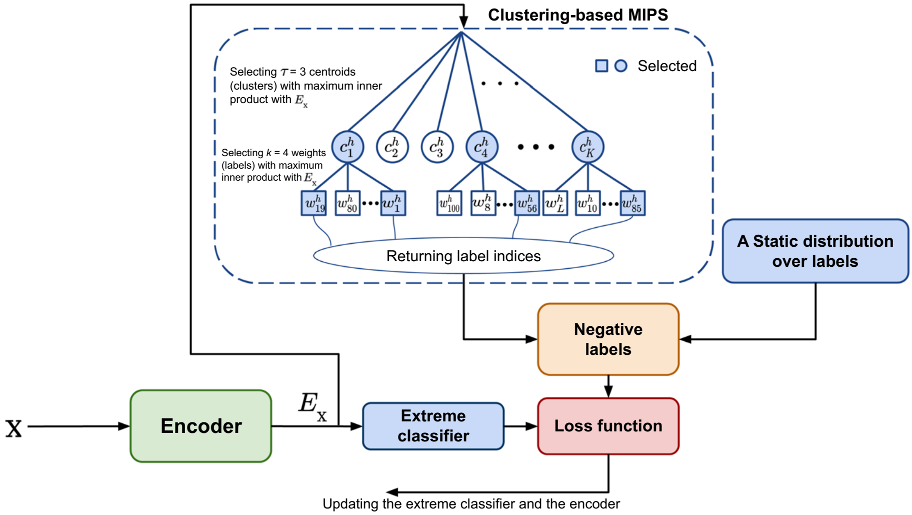

Clustering-based MIPS
Selecting τ = 3 centroids (clusters) with maximum inner product with E_x
□ ○ Selected
Selecting k = 4 weights (labels) with maximum inner product with E_x
c_1^h
c_2^h
c_3^h
c_4^h
c_K^h
w_{19}^h
w_{80}^h
w_1^h
w_{100}^h
w_8^h
w_{56}^h
w_L^h
w_{10}^h
w_{85}^h
Returning label indices
A Static distribution over labels
Negative labels
x
Encoder
E_x
Extreme classifier
Loss function
Updating the extreme classifier and the encoder

Fig. 4 A forward pass and the loss computation in our proposed negative sampling method which restricts the negative part of the loss to hard negatives suggested by a clustering-based MIPS as well as negative labels sampled from a static distribution

where τ is a hyperparameter which indicates how many clusters to be searched. Let $I^* = \bigcup_{i \in C^*} I_i$, the next step is to search the weights suggested by I^* as follows:

$$L_M = k\text{-}\underset{i \in I^*}{\operatorname{argmax}}\, E_\mathrm{x}^T \mathrm{w}_i^h \tag{7}$$

where w_i^h is the i-th column of W^h and k is the number of labels retrieved by the approximate MIPS. The last step for using the labels suggested by MIPS as negative labels is to exclude positive labels from L_M:

$$L_M \leftarrow L_M \setminus \{j | y_j = 1\} \tag{8}$$

Finally, we combine the hard negatives suggested by the MIPS with several labels drawn from a static distribution. The static distribution should be model-independent and fixed during training to be possible to easily sample from it, for which we use a uniform distribution in our experiments. Assume q is a uniform distribution defined over the set of negative labels, and the set L_S denotes the labels drawn from q. The negative part of the loss in Eq. 2 is computed only over the union of L_M and L_S:

$$\hat{l}(f(E_\mathrm{x}), \mathrm{y}) = \sum_{j:y_j=1} l_+(f_j(E_\mathrm{x})) + \sum_{j \in L_M \cup L_S} l_-(f_j(E_\mathrm{x})) \tag{9}$$

The proposed negative sampling method is summarized in Algorithm 1 as well as Fig. 4.

(Dis)similarities with meta-classifier-based methods Looking at Fig. 4, one may notice similarities between the proposed method and meta-classifier-based methods such as LightXML (Jiang et al., 2021) and AttentionXML (You et al., 2019). Both approaches use clustering to find hard negatives for the extreme classifier. More precisely, first, the clusters

```
Input: Data $\{x_i, y_i\}_{i=1}^{N}$, Encoder with parameters $\theta$, Extreme classifier with parameters $W$, Number of negative labels $R$, A uniform distribution over labels $q$, Maximum iterations $T$, Preprocessing intervals $P$, Number of clusters $K$, Clusters to be searched $\tau$, Number of negative labels from MIPS $k$
$t \leftarrow 0$
while $t < T$ do
    if $t\%P = 0$ then
        $C^h \leftarrow$ cluster columns of $W$ into $K$ partitions and compute their centroids using spherical $K$-means
        $W^h \leftarrow W$
    end if
    $x, y \leftarrow$ sample one instance and corresponding labels from $\{x_i, y_i\}_{i=1}^{N}$ ▷ can also be extended to a minibatch
    $E_x \leftarrow$ encode x by the encoder
    $C^* \leftarrow \tau\text{-}\operatorname{argmax}_{c \in C^h} E_x^T c$
    $I^* \leftarrow$ union of the labels in the clusters in $C^*$
    $L_M \leftarrow k\text{-argmax}_{i \in I^*} E_x^T w_{*i}^h$
    $L_M \leftarrow L_M \setminus \{j | y_j = 1\}$
    $L_S \leftarrow$ sample $R - k$ labels from $q$
    $L_S \leftarrow L_S \setminus \{j | y_j = 1\}$
    Compute the loss using Equation 9
    Update $\theta$ and those columns of $W$ corresponding to labels included in the loss
    $t \leftarrow t + 1$
end while
return None
```

Algorithm 1 Proposed negative sampling method

are scored based on their similarities with the encoded samples. Then, the union of the labels in the clusters with the highest scores are used as negative labels for the extreme classifier. The module which is responsible for finding hard negatives is called a meta classifier in meta-classifier-based methods, while it is denoted as clustering-based MIPS in our method in Fig. 4.

Despite similarities between the forward pass of the two approaches, there are three main differences between our method and meta-classifier-based methods. (I) In meta-classifier-based methods, the representations of the labels[1] for clustering are independent of the extreme classifier, which means that no matter how accurate the meta classifier is, the labels suggested by the meta classifier may not be the hardest ones for the extreme classifier. However, in our method, the representations of the labels are completely aligned with the extreme classifier as they are the columns of the weight matrix W^h. (II) As pointed out in Kharbanda et al. (2021), since the meta and extreme classifiers

[1] In most of meta-classifier-based XMC models, each label is simply represented by aggregating the TF-IDF representations of the relevant instances for that label.

operate on different resolutions, it may be difficult to learn data representations which are simultaneously suitable for both tasks in the meta-classifier-based methods. On the other hand, our scheme only requires intermittent re-clustering of the weight vectors. (III) Also, in meta-classifier-based methods, the meta classifier needs to be trained and stored in addition to the extreme classifier, which can add millions of more parameters to the overall model. While in our method, no extra space, other than the extreme classifier, is required.

Complexity and hyperparameters: There are three hyperparameters involved in the MIPS part of our method: K, τ, k, which are the total number of clusters, the number of clusters to be searched, and the number of labels retrieved by the MIPS, respectively. Among these hyperparameters, K and τ have a significant role in the number of computations. Assuming the clusters are balanced, a large K will lead to centroids which are better representatives of the parameters, but it also increases the computations for comparing the embeddings with the centroids. Using a small K decreases those computations but will lead to having large cluster sizes which will again lead to high computations when the clusters need to be searched. As stated in Zhang et al. (2021), $K = \sqrt{L}$ is a good choice, which is also used in all the experiments in our work. In this case, with the assumption that the clusters are balanced, the computational complexity of querying the MIPS will be $\mathcal{O}(\tau\sqrt{L}d)$. If $\tau = \sqrt{L}$, which is equal to the number of clusters, then the computational complexity of the MIPS will be the same as an exhaustive search, while the lowest computational complexity is for the case $\tau = 1$. The rest of the computations in the forward pass will be only drawing labels from the predefined distribution and computing the loss over them, which is the same as static negative sampling methods.

Since the preprocessing step of the MIPS is done only within large intervals, the computational complexity of this step is negligible compared to the whole training time.

In the backward pass, the number of parameters needed to be updated is exactly proportional to the number of negative labels used in computing the loss. This is contrary to meta-classifier-based methods, like LightXML, in which, in addition to the extreme classifier, all the parameters of a large meta classifier should be updated at each iteration of the training algorithm.

5 Experiments

In this section, we evaluate the proposed negative sampling method of Sect. 4, which combines a few negative labels suggested by a clustering-based MIPS with several labels sampled from a uniform distribution. The purpose of the experiments is twofold: firstly, to compare the proposed method with the case that all the negative labels are selected by MIPS, and secondly, to compare it with meta-classifier-based negative sampling methods, for which we use the LightXML framework of Jiang et al. (2021), and see if the proposed adaptive negative sampling can reach the performance of meta-classifier-based methods.

The results show that the proposed method achieves significantly higher precision compared to the case that all the negative labels are selected by MIPS or the case that the labels are sampled only from a uniform distribution. Moreover, the proposed method consistently achieves high precision even in the presence of large intervals for pre-processing the weights, while in this setup, it is difficult to train the model using only the negative labels suggested by the MIPS module. Compared to LightXML, while both the

Table 1 The statistics of three datasets used in our experiments (Bhatia et al., 2016)

Dataset	#Training	#Test	#Labels	APpL	ALpP
Amazon-670K	490,449	153,025	670,091	3.99	5.45
WikiLSHTC-325K	1,778,351	587,084	325,056	17.46	3.19
Wikipedia-500K	1,813,391	783,743	501,070	24.75	4.77

APpL indicates the average points per label and ALpP is the average labels per point

proposed method and LightXML have very similar architectures, the proposed method can achieve precision near to that of LightXML with significantly smaller model size and less training time.

5.1 Setup

Architectures and hyperparameters: We use two architectures as the encoders for the proposed method (and the corresponding baselines): a shallow neural network with a single hidden layer and a BERT model.

In the case of a shallow neural network as the encoder, a dropout with $p = 0.2$ is used over the input features, the number of neurons in the hidden layer is set to 128, ReLU is the activation function of the neurons in the hidden layer, the optimizer is Adam and the learning rate is chosen according to a validation set split from the training data. Also, the number of negative labels for approximating the loss is around 1% of the total number of labels.[2] In the MIPS-based negative sampling methods, we perform pre-processing of the weights only once per epoch.

For the BERT model as the encoder, all the hyperparameters, including the number of negative labels for approximating the loss in the extreme classifier and the learning rate, are the same as those used by Jiang et al. (2021). In MIPS-based negative sampling methods for training the BERT model, we preform pre-processing of the weights after each 1000 iterations.

For both the shallow neural network and the BERT model, the number of negative labels retrieved by the MIPS procedure in the MIPS-based methods is 5, which is approximately equal to the average number of positive labels per data point in the datasets we used in our experiments (Table 1). Also, for both models, the loss function is the BCE loss that is used.

We use the GPU-implemented MIPS of Johnson et al. (2019) for performing maximum inner product search in the MIPS-based negative sampling methods. Two hyperparameters related to the MIPS are K, which is the number of clusters, and τ, which is the number of clusters to be searched during querying. In our experiments, K is always set to the square root of the number of labels, as discussed in Sect. 4, and τ is set to 64.

For evaluating the BERT models trained by MIPS-based methods, to avoid performing an exhaustive search over all the labels, we approximate the labels with the highest scores using an approximate MIPS. Contrary to the training phase, for evaluation, the pre-processing of weights needs to be done only once as the weights are fixed. Similar to the

[2] The number of negative labels in negative sampling methods is determined based on time and space constraints. In our experiments with shallow neural networks, we set it to 1% of the total number of labels, which is close to those employed in LightXML.

training phase, for evaluation using MIPS, K is set to the square root of the number of labels and τ is chosen in a way that the evaluation time be the same as that of LightXML.

Baselines: We compare the proposed method with some other strategies for negative sampling. These methods are as follows:

- `Uniform`: This method samples negative labels uniformly from the set of all possible negative labels.
- `MIPS-s`: This method refers to the proposed method, in which only a few negative labels are selected by the clustering-based MIPS and the rest are sampled uniformly.
- `MIPS-a`: In this method, after training by only uniform negative sampling for a couple of iterations, all the needed negative labels selected by MIPS.
- `LightXML`: This method refers to the meta-classifier-based negative sampling proposed by Jiang et al. (2021).

Datasets: We use three textual multilabel datasets from the extreme classification repository (Bhatia et al., 2016). The statistics of these datasets are given in Table 1. For the shallow neural network, we use the TF-IDF representations of the data, while for the BERT model, the raw texts are used. Since the raw text is not available for WikiLSHTC-325K, the experiments using BERT were only done on the two other datasets.

Evaluation metrics: In the multilabel setting of our experiments, the goal is to predict the correct labels among the top-k labels. Therefore, we evaluate the models using precision at k (P@k) and its unbiased counterpart, propensity scored precision at k (PSP@k). Formally, these two metrics are as follows:

$$\mathrm{P@k}(\mathrm{y}, \hat{\mathrm{y}}) := \frac{1}{k} \sum_{l \in topk(\hat{\mathrm{y}})} y_l \tag{10}$$

$$\mathrm{PSP@k}(\mathrm{y}, \hat{\mathrm{y}}) := \frac{1}{k} \sum_{l \in topk(\hat{\mathrm{y}})} \frac{y_l}{p_l} \tag{11}$$

where $topk(\hat{\mathrm{y}})$ is the set of top-k labels predicted by the model, and p_l is the propensity score of label l, which indicates the probability that label l is present (Jain et al., 2016). The propensity scores for each dataset are set according to Jain et al. (2016).

5.2 Results on the shallow neural network as the encoder

In this subsection, we compare the proposed method with the baselines using the shallow neural network. The results are given in Table 2. As the results show, all the metrics for `MIPS-s` are significantly higher than those of `MIPS-a` and `Uniform`, which shows that using only a few labels suggested by MIPS combined with randomly sampled labels can boost the performance significantly compared to using only labels from MIPS or only uniformly sampled labels.

Also, we should note that the results for the MIPS-a method without the initial training by uniform negative sampling are not higher than a random guess even with very high values for τ and very short intervals for preprocessing the weights, which makes it close to the setting discussed in Sect. 3.1.

Table 2 A comparison of the proposed method (MIPS-s) with uniform negative sampling (Uniform) and the case that all the needed negative labels are defined by MIPS (MIPS-a) for training a neural network with one hidden layer on three extreme classification datasets

Dataset	Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
Amazon-670K	Uniform	35.0	31.2	28.3	20.2	22.8	25.1
	MIPS-a	37.1	33.2	30.3	23.3	25.9	28.1
	MIPS-s	40.2	35.8	32.6	26.2	29.0	31.3
WikiLSHTC-325K	Uniform	52.5	33.9	25.5	20.6	24.4	27.8
	MIPS-a	54.5	34.6	25.5	23.2	25.9	28.4
	MIPS-s	59.4	37.5	27.3	26.6	30.0	32.7
Wikipedia-500K	Uniform	50.0	33.6	26.2	16.8	19.3	21.8
	MIPS-a	53.8	35.6	27.7	19.6	21.8	24.3
	MIPS-s	62.3	41.2	31.1	24.7	27.5	29.5

The results show that in all the cases, the proposed method surpasses the aforementioned baselines

Table 3 A comparison of the proposed method (MIPS-s) with uniform negative sampling (Uniform), the case that all the needed negative labels are defined by the MIPS (MIPS-a), and LightXML for training a BERT model on two extreme text classification datasets

Dataset	Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
Amazon-670K	Uniform	39.2	35.4	32.5	19.4	23.5	27.1
	MIPS-a	45.9	41.0	37.0	25.0	29.6	33.4
	MIPS-s	46.1	41.0	37.2	26.3	30.5	34.1
	LightXML	46.8	41.7	37.6	27.0	31.2	34.6
Wikipedia-500K	Uniform	48.8	35.5	28.7	16.8	20.5	23.9
	MIPS-a	72.5	54.4	42.2	27.2	33.7	37.8
	MIPS-s	75.5	55.7	42.5	31.9	38.2	41.2
	LightXML	74.9	56.0	43.1	28.7	35.6	39.4

The results show that, similar to the results using the shallow neural network, the proposed method surpasses Uniform and MIPS-a. Also, the proposed method achieves precision near to that of LightXML

5.3 Results on BERT as the encoder

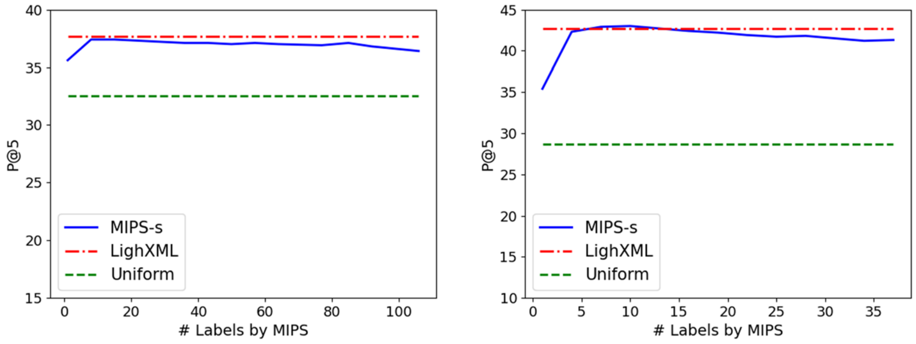
Table 3 compares different negative sampling methods for training BERT on Amazon-670K and Wikipedia-500K. As the results show, the MIPS-s method outperforms Uniform on both datasets. Compared to MIPS-a, the MIPS-s method achieves superior performance across the majority of the metrics on Wikipedia-500K. While on Amazon-670K both methods exhibit similar efficacy, in the next subsection, we show sensitivity of the MISP-a method to the pre-processing intervals of the MIPS module. Specifically, we show MIPS-a fails to train properly in the presence of larg intervals for pre-processing the weights, highlighting the limitations of this approach.

Compared to LightXML, precision and ps-precision achieved by MIPS-s are within 1% of that obtained by LightXML. Unfortunately, given the large scale of out datasets, it is difficult to verify the statistical significance of the results. However, we ran MIPS-s and LightXML on Amazon-670K using four different sets of initial weights, the detail of which

Table 4 A comparison of the model size and training time of MIPS-s with those of LightXML on Amazon-670K and Wikipedia-500K

Dataset	Method	Model size (GB)	Training time (h)
Amazon-670K	LightXML	1.6	34.2
	MIPS-s	1.2	29.5
Wikipedia-500K	LightXML	1.5	45.0
	MIPS-s	1.0	43.7

The results show the MIPS-smethod can save almost 33% space for storing the model and leads to lower training time



(a) Amazon-670K (b) Wikipedia-500K

Fig. 5 The effect of the number of negative labels retrieved by the MIPS in MIPS-s on P@5. In both datasets, P@5 peaks when the number of negative labels retrieved by the MIPS is almost 10. Furthermore, the figures show that the MIPS-s method at it's peak can achieve P@5 near to that of LightXML. Also, there is a large gap between MIPS-s and Uniform even when only 1 negative label is taken from the MIPS

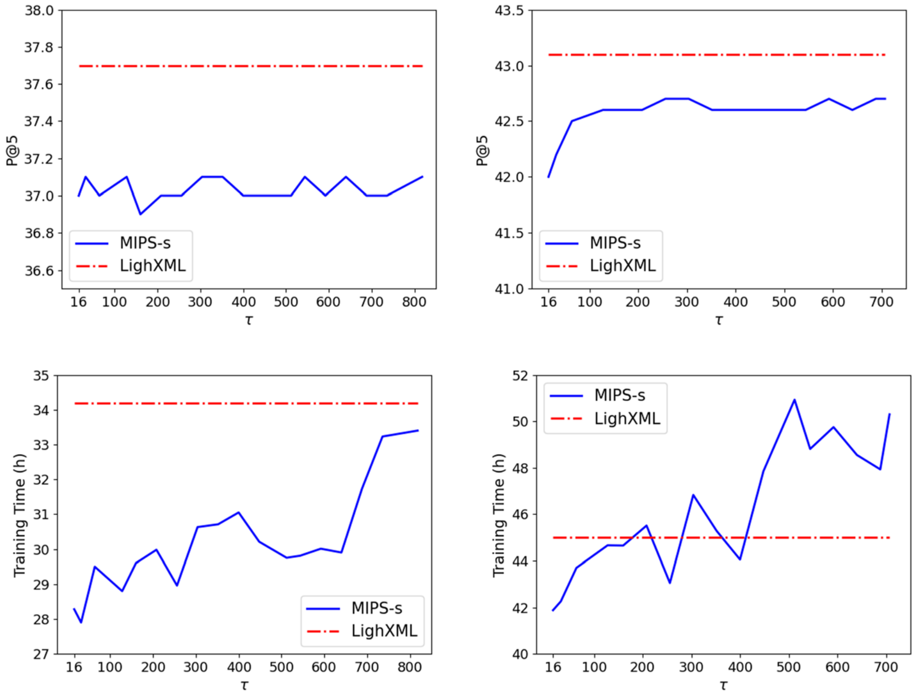
are given in Table 5 in Appendix A. The results show the standard deviation of all the metrics for both MIPS-s and LightXML are less than 0.07% and 0.08%, respectively, which indicates that the performance of these models shows minimal variance on the Amazon-670K dataset, suggesting a consistent performance.

We should note that similar to the experiments on the shallow neural network, when the negative labels are restricted to the hardest ones from the beginning of training in the BERT model, the output is not better than a random guess.

The main difference between the architecture of LightXML and the proposed method is that there is no need to train and store any meta classifier in the latter one. Table 4 compares the model size of MIPS-s with LightXML as well as the training time. The results show by excluding the meta classifier in LightXML, we can save up to 34% space on the hard disk and train the model a few hours faster.

5.4 Hyperparameters analysis

In this subsection, we investigate the effects of three hyperparameters, namely the number of negative labels taken from the MIPS module, the number of clusters to be searched (τ), and pre-processing intervals of the MIPS module. All the experiments are done using Bert



(a) Amazon-670K (b) Wikipedia-500K

Fig. 6 The effect of τ (the number of clusters to be searched in the MIPS method during training) on P@5 (top row) and training time (bottom row) in the `MIPS-s` method. Even small values for τ can achieve high P@5. Increasing τ will increase the training time, while there is no significant gain in P@5

as the encoder. Other hyperparameters than the ones which are being analysed are the same as Sect. 5.3.

Number of negatives taken from the MIPS: Fig. 5 compares P@5 in the `MIPS-s` method when the number of negative labels retrieved by the MIPS is different. The results show that P@5 reaches its peak when the number of negative labels obtained by the MIPS module is around 10. Also, it can be seen in the figure that the maximum P@5 achieved by `MIPS-s` is very close to `LightXML`. Moreover, even using one label found by the MIPS can boost the performance significantly when the results are compared with `Uniform`.

We should note that since the number of labels taken from the MIPS module has minimal impact on the training time in `MIPS-s`, the training times of all the experiments with `MIPS-s` in Fig. 5 are approximately the same as that mentioned for `MIPS-s` in Table 4.

The number of clusters to be searched (τ): Fig. 6 demonstrates P@5 and training time with different values for τ during training `MIPS-s`. Smaller values for τ indicate a smaller search space and, therefore, faster training, while larger values approach an exact search, which is computationally costly. However the figure shows that even with small values for τ, `MIPS-s` can achieve high P@5, with the difference of only around 1% compared to LightXML, while increasing τ only marginally improves P@5.

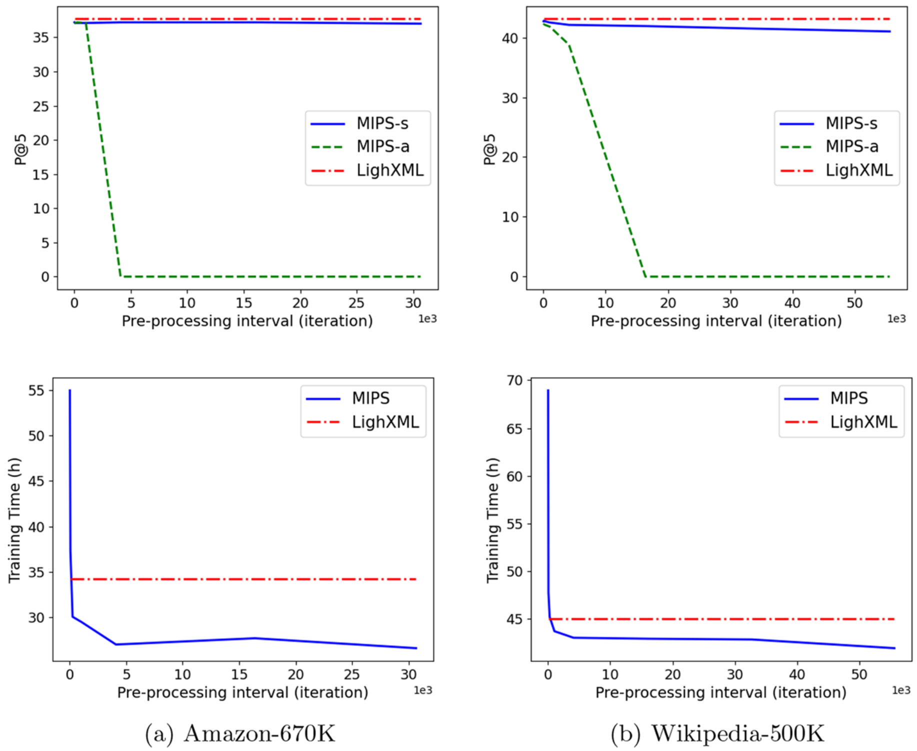


Fig. 7 The effect of pre-processing intervals of the MIPS module in MIPS-based methods on P@5 (top row) and training time (bottom row). Since both MIPS-based methods have approximately similar training times, only the training time of `MIPS-s` is reported. Both `MIPS-s` and `MIPS-a` methods achieve high P@5 when the intervals for pre-processing are short and therefore the training time is high. However, the `MIPS-a` method fails to train properly when the pre-processing intervals are large, while `MIPS-s` consistently achieves high P@5

Pre-processing intervals: Fig. 7 shows the effect of the length of pre-processing intervals of the weights in the MIPS module in MIPS-based methods on P@5 and training time. Consistent with the analysis of Sect. 3.2, while `MIPS-s` achieves high P@5 even with large pre-processing intervals (more than 30K on Amazon-670K and 50K on Wikipedia-500K), the `MIPS-a` method can be trained properly only in the high training time region with short intervals for pre-processing.

In addition to the above hyperparameters, we also investigated the effect of using the conventional K-means equipped with Euclidean distance instead of spherical K-means on our proposed method. The results presented in Table 6 in Appendix A show that both clustering methods achieve similar performance highlighting the robustness of our approach to the clustering method.

6 Other related work

In this section, we review some related work on adaptive and meta-classifier-based negative sampling methods as well as the work on the drawbacks of hard negative mining.

MIPS-based adaptive negative sampling—In MIPS-based extreme classification, to find hard negative labels, maximum inner product search methods are utilized to approximate the labels which maximize the inner product between an embedding and the labels' weights (Chen et al., 2020; Daghaghi et al., 2021; Vijayanarasimhan et al., 2014; Yen et al., 2018). Among these works, Daghaghi et al. (2021) proposed to use Local Sensitive Hashing (LSH) for performing the maximum inner product search. Yen et al. (2018), to reduce the computational complexity, replaced the MIPS over the high dimensional embeddings with several lower dimensional ones by decoupling the dimensions using dual decomposition. Chen et al. (2020) proposed to approximate the neurons with high responses in every layer of neural networks using an approximate MIPS and backpropagate the error only through those neurons.

Meta-classifier-based negative sampling—In meta-classifier-based methods, a meta classifier is trained in addition to the extreme classifier to suggest the confusing labels to the extreme classifier. In the recent state-of-the-art meta-classifier-based methods, first, the labels are categorized into a bunch of clusters based on their similarities. Then, a meta classifier is trained in which for any sample, any cluster containing at least one positive label is considered positive and the rest negative labels. Finally, the meta classifier is queried for each sample and the union of the labels in a few clusters with the highest scores are used as the negative labels for approximating the loss of the extreme classifier.

Among these works, You et al. (2019) and Zhang et al. (2021) use hierarchical architectures where one model is trained at each level which acts as the meta classifier for the model at the next level. In these methods, the models are trained sequentially, which means the hard negatives retrieved by their meta-classifier are fixed during training. To make the process of the learning the embeddings in the meta classifier more aligned to the extreme task, Kharbanda et al. (2021) propose to increase the number of clusters for the meta-classfier.

Two-stage negative sampling—Two-stage negative sampling methods have recently become the state-of-the-art in short-text extreme classification (Dahiya et al., 2021a, 2021b, 2022; Mittal et al., 2021). In most of these methods, in the first stage, a model is trained to learn representations for the documents and the labels. Then in the second stage, by using an approximate nearest neighbor search over the learned embeddings, labels with small distances to each instance are determined and be used as hard negatives for training the extreme classifier.

Two-stage negative sampling methods are at the intersection of adaptive methods and meta-classifier-based ones since they are partially dependent on the model and also they use the model in the first stage as a means for finding confusing labels for the extreme classifier.

Issues in training using hard negatives—A few works have discussed issues in training using hard negative labels (Schroff et al., 2015; Wu et al., 2017). Specifically, in deep embedding learning using triplet losses, (Schroff et al., 2015) argued that selecting the hardest negative, which is the sample in the mini-batch with the closest distance to the anchor, will lead to a bad local minimum in the early stages of training. Wu et al. (2017) showed that in these methods, the gradient with respect to hard negatives have high variance and therefore susceptible to the noise in the training data. To this end, they proposed a distance weighted sampling, in which the probability of selecting a hard sample is equal to a constant and for the rest of the labels it is proportional to the inverse of the distance to the anchor. Our work is on the same line as the aforementioned works to investigate the issues related to using negative labels in MIPS-based adaptive methods to train feedforward neural networks.

7 Conclusion

In this paper, we highlighted two difficulties in training deep neural networks using MIPS-based adaptive negative sampling methods. We argued that when only hard negatives are employed from the beginning of training, the gradients of the loss with respect to the embeddings are unstable. Also, we showed that when the labels needed for approximating the loss are only determined by MIPS, training is sensitive to the pre-processing intervals of the weights in the MIPS method.

To overcome these problems, we proposed to only pick a few labels from those found by MIPS and select the rest from a uniform distribution. The results show that the proposed method leads to significantly higher precision in a shallow and a deep neural network. Furthermore, we highlighted the similarities of the architecture of our method with LightXML, a meta-classifier-based negative sampling. We showed that our approach, which only requires re-clustering the weights a few times during training, can reach the performance of LightXML on a BERT model, while there is no need to train and store any additional classifier. We hope that our work will spur further research in exploring meta-classifier free negative sampling methods in extreme multi-label classification.

Appendix A: Supplementary experiments

See Tables 5 and 6.

Table 5 Results of MIPS-s and LightXML on Amazon-670K using four different initial weights

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
MIPS-s	46.2	41.1	37.2	26.4	30.6	34.1
	46.1	41.1	37.2	26.3	30.6	34.1
	46.1	41.0	37.2	26.3	30.5	34.1
	46.0	41.1	37.2	26.3	30.6	34.1
Mean ± SD	46.1 ± 0.07	41.0 ± 0.04	37.2 ± 0.00	26.3 ± 0.04	30.6 ± 0.04	34.1 ± 0.00
LightXML	46.7	41.7	37.7	26.9	31.3	34.7
	46.7	41.6	37.5	26.9	31.1	34.5
	46.8	41.7	37.6	27.0	31.3	34.7
	46.8	41.6	37.6	27.0	31.2	34.6
Mean ± SD	46.8 ± 0.05	41.7 ± 0.05	37.6 ± 0.07	27.0 ± 0.05	31.2 ± 0.08	34.6 ± 0.08

The results show that the standard deviation of all the metrics for both MIPS-s and LightXML are less than 0.07% and 0.08%, respectively, suggesting a consistent performance

Table 6 A comparison of standard K-means based on Euclidean distance and spherical K-means for pre-processing the weights in MIPS-s

Dataset	K-means variant	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
Amazon-670K	Standard	46.0	40.9	37.0	26.3	30.4	33.9
	Spherical	46.1	41.0	37.2	26.3	30.5	34.1
Wikipedia-500K	Standard	75.3	55.6	42.4	32.2	38.3	41.3
	Spherical	75.5	55.7	42.5	31.9	38.2	41.2

The results demonstrate the comparable performance of spherical K-means over the standard one

Author contributions MQ and RB proposed the idea. MQ did the literature review, designed and implemented the methods, ran the experiments, and wrote the first draft. RB supervised the project with continuous discussions and contributed towards writing and reviewing the manuscript.

Funding Open Access funding provided by Aalto University. Open Access funding provided by Aalto University. The authors would like to acknowledge research funding from the Academy of Finland projects (347707 and 348215).

Availability of data and materials The datasets used in our experiments are publicly available in the extreme classification repository http://manikvarma.org/downloads/XC/XMLRepository.html.

Code availability Our code for the experiments is available at https://github.com/xmc-aalto/mips-negative-sampling.

Declarations

Conflict of interest The authors declare no conflict of interest to disclose associated with this article.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication The authors give consent for the publication of identifiable details in this paper, including figures, tables, and the results, in other projects by mentioning the reference.

References

Auvolat, A., Chandar, S., Vincent, P., Larochelle, H., & Bengio, Y. (2015). Clustering is efficient for approximate maximum inner product search. arXiv preprint arXiv:1507.05910 .

Babbar, R., & Schölkopf, B. (2017). Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM international conference on web search and data mining*, (pp. 721–729).

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

Bengio, Y., & Senécal, J. S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks, 19*(4), 713–722.

Bhatia, K., Dahiya, K., Jain, H., Kar, P., Mittal, A., Prabhu Y., & Varma, M. (2016). *The extreme classification repository: Multi-label datasets and code*.

Blanc, G., & Rendle, S. (2018). Adaptive sampled softmax with kernel based sampling. In *International conference on machine learning*, (pp. 590–599). PMLR.

Chen, B., Medini, T., Farwell, J., Tai, C., Shrivastava, A., et al. (2020). Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems, 2*, 291–306.

Daghaghi, S., Medini, T., Meisburger, N., Chen, B., Zhao, M., & Shrivastava, A. (2021). A tale of two efficient and informative negative sampling distributions. In *International conference on machine learning*, (pp. 2319–2329). PMLR.

Dahiya, K., Agarwal, A., Saini, D., Gururaj, K., Jiao, J., Singh, A., Agarwal, S., Kar, P., Varma, M. (2021a). Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International conference on machine learning*, (pp. 2330–2340). PMLR.

Dahiya, K., Gupta, N., Saini, D., Soni, A., Wang, Y., Dave, K., & Varma, M. (2022). *Ngame: Negative mining-aware mini-batching for extreme classification*. arXiv preprint arXiv:2207.04452 .

Dahiya, K., Saini, D., Mittal, A., Shaw, A., Dave, K., Soni, A., & Varma, M. (2021b). Deepxml: A deep extreme multi-label learning framework applied to short text documents. In *Proceedings of the 14th ACM international conference on web search and data mining*, (pp. 31–39).

Jain, H., Balasubramanian, V., Chunduri, B., & Varma, M. (2019). Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the twelfth ACM international conference on web search and data mining*, (pp. 528–536).

Jain, H., Prabhu, Y., Varma, M. (2016). August. Extreme multi-label loss functions for recommendation, tagging, ranking and other missing label applications. In *KDD*.

Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z., & Zhuang, F. (2021). Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. *Proceedings of the AAAI Conference on Artificial Intelligence, 35*, 7987–7994.

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUS. *IEEE Transactions on Big Data, 7*(3), 535–547.

Khandagale, S., Xiao, H., & Babbar, R. (2020). Bonsai: Diverse and shallow trees for extreme multi-label classification. *Machine Learning, 109*, 1–21.

Kharbanda, S., Banerjee, A., Gupta, D., Palrecha, A., & Babbar, R. (2023). Inceptionxml: A lightweight framework with synchronized negative sampling for short text extreme classification. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 760–769).

Kharbanda, S., Banerjee, A., Schultheis, E., Babbar, R. (2022). Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. In *Advances in neural information processing systems*.

Lee, K., Chang, M. W., & Toutanova, K. (2019). *Latent retrieval for weakly supervised open domain question answering*. arXiv preprint arXiv:1906.00300 .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Mittal, A., Dahiya, K., Agrawal, S., Saini, D., Agarwal, S., Kar, P., & Varma, M. (2021). Decaf: Deep extreme classification with label features. In *Proceedings of the 14th ACM international conference on web search and data mining*, (pp. 49–57).

Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M. R., & Galinari, P. (2015). Lshtc: A benchmark for large-scale text classification. arXiv preprint arXiv:1503.08581.

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., Varma, M. (2018). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, (pp. 993–1002).

Rawat, A. S., Menon, A.K., Jitkrittum, W., Jayasumana, S., Yu, F., Reddi, S., & Kumar, S. (2021). Disentangling sampling and labeling bias for learning in large-output spaces. In *International conference on machine learning*, (pp. 8890–8901). PMLR.

Rawat, A. S., Chen, J., Yu, F. X. X., Suresh, A. T., & Kumar, S. (2019). Sampled softmax with random fourier features. *Advances in Neural Information Processing Systems, 32*, 13857–13867.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 815–823).

Schultheis, E., & Babbar, R. (2022). Speeding-up one-versus-all training for extreme classification via mean-separating initialization. *Machine Learning, 111*, 1–24.

Shrivastava, A., & Li, P. (2014). Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in neural information processing systems*, *27*.

Vijayanarasimhan, S., Shlens, J., Monga, R., Yagnik, J. (2014). *Deep networks with large output spaces*. arXiv preprint arXiv:1412.7479.

Wu, C. Y., Manmatha, R., Smola, A. J., & Krahenbuhl, P. (2017). Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, (pp. 2840–2848).

Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P., Ahmed, J., & Overwijk, A., (2020). *Approximate nearest neighbor negative contrastive learning for dense text retrieval*. ICLR .

Yen, I. E. H., Kale, S., Yu, F., Holtmann-Rice, D., Kumar, S., & Ravikumar, P., (2018). Loss decomposition for fast learning in large output spaces. In *International Conference on Machine Learning*, (pp. 5640–5649). PMLR.

You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H., & Zhu, S. (2019). Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *NeurIPS*, (pp. 5812–5822).

Zhang, J., Chang, W. C., Yu, H. F., & Dhillon, I. (2021). Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems, 34*, 7267–7280.

Zhong, S. (2005). Efficient online spherical k-means clustering. *Proceedings 2005 IEEE International Joint Conference on Neural Networks, 5*, 3180–3185.