NGAME: Negative Mining-aware Mini-batching for Extreme Classification

Kunal Dahiya*
kunalsdahiya@gmail.com
IIT Delhi
India

Nilesh Gupta*†
nileshgupta2797@utexas.edu
UT Austin
USA

Deepak Saini*
desaini@microsoft.com
Microsoft
USA

Akshay Soni
akson@microsoft.com
Microsoft
USA

Yajun Wang
yajwang@linkedin.com
Linkedin
USA

Kushal Dave
Jian Jiao
kudave@microsoft.com
jiajia@microsoft.com
Microsoft
USA

Gururaj K
Prasenjit Dey
Amit Singh
gururajk@microsoft.com
prdey@microsoft.com
siamit@microsoft.com
Microsoft
India

Deepesh Hada
Vidit Jain
Bhawna Paliwal
deepeshhada@microsoft.com
jainvidit@microsoft.com
bhawna@microsoft.com
Microsoft Research
India

Anshul Mittal
me@anshulmittal.org
IIT Delhi
India

Sonu Mehta
Sonu.Mehta@microsoft.com
Microsoft Research & IIT Delhi
India

Ramachandran Ramjee
ramjee@microsoft.com
Microsoft Research
India

Sumeet Agarwal
sumeet@ee.iitd.ac.in
IIT Delhi
India

Purushottam Kar†
purushot@cse.iitk.ac.in
IIT Kanpur
India

Manik Varma
manik@microsoft.com
Microsoft Research
India

ABSTRACT

Extreme Classification (XC) seeks to tag data points with the most relevant subset of labels from an extremely large label set. Performing *deep XC* with dense, learnt representations for data points and labels has attracted much attention due to its superiority over earlier XC methods that used sparse, hand-crafted features. Negative mining techniques have emerged as a critical component of all deep XC methods, allowing them to scale to millions of labels. However, despite recent advances, training deep XC models with large encoder architectures such as transformers remains challenging. This paper notices that memory overheads of popular negative mining techniques often force mini-batch sizes to remain small and slow training down. In response, this paper introduces NGAME, a light-weight mini-batch creation technique that offers provably accurate in-batch negative samples. This allows training with larger mini-batches offering significantly faster convergence and higher accuracies than existing negative sampling techniques. NGAME was found to be up to 16% more accurate than state-of-the-art methods on a wide array of benchmark datasets for extreme classification, as well as 3% more accurate at retrieving search engine queries in response to a user webpage visit to show personalized ads. In live A/B tests on a popular search engine, NGAME yielded up to 23% gains in click-through-rates. Code for NGAME is available at https://github.com/Extreme-classification/ngame

CCS CONCEPTS

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

*These authors contributed equally
†Work done while the author was at Microsoft

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570392

KEYWORDS

Extreme multi-label learning; large-scale learning; negative sampling; Siamese networks; sponsored search; personalized ads

ACM Reference Format:

Kunal Dahiya, Nilesh Gupta, Deepak Saini, Akshay Soni, Yajun Wang, Kushal Dave, Jian Jiao, Gururaj K, Prasenjit Dey, Amit Singh, Deepesh Hada, Vidit Jain, Bhawna Paliwal, Anshul Mittal, Sonu Mehta, Ramachandran Ramjee, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2023. NGAME: Negative Mining-aware Mini-batching for Extreme Classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570392

1 INTRODUCTION

Overview: Extreme Classification (XC) requires predicting the most relevant *subset* of labels for a data point from an extremely large set of labels. Note that multi-label classification generalizes multi-class classification which instead aims to predict a single label from a mutually exclusive label set. In recent years, XC has emerged as a workhorse for several real-world applications such as product recommendation [14, 36, 38], document tagging [2, 9, 60], search & advertisement [14, 24, 43], and query recommendation [9, 23].

Challenges in XC: XC tasks give rise to peculiar challenges arising from application-specific demands as well as data characteristics. For instance, to adequately serve real-time applications including ads recommendation, XC routines must offer millisecond-time inference, even if selecting among several millions of labels. Training models at extreme scales is similarly challenging due to the infeasibility of training on all data point-label pairs when the number of data points and labels both are in the millions. This necessitates the use of some form of *negative mining* technique wherein a data point is trained only with respect to its positive labels (of which there are usually a few) and a small set of carefully chosen negative labels. Training is made even more challenging due to the abundance of *rare* or *tail* labels for which few, often less than 10, training points are available. It is common for a majority of labels to be rare in retrieval, recommendation, and tagging tasks.

Deep Siamese XC: The recent years have seen XC models become more and more accurate with the development of two specific design choices. Firstly, XC methods identified the benefits of using label metadata in various forms such as textual descriptions of labels [9, 13, 38] or label correlation graphs [40, 51]. This is in sharp contrast to earlier work in XC that treated labels as featureless identifiers (please see Section 2 for a survey). Secondly, XC methods began reaping the benefits of deep-learnt embeddings by using deep encoder architectures to embed both data points and labels. This was yet another departure from traditional XC methods that relied on sparse, hand-crafted features such as bag-of-words. In particular, recent work has demonstrated advantages of using *Siamese* architectures wherein representations for both data points as well as labels are obtained using a shared embedding model [13, 33, 38]. Some of these techniques [13, 38] demonstrate further gains in accuracy by fine-tuning the label representation offered by the shared embedding architecture to yield label classifiers. Other methods such as [61] focus on training large transformer models at extreme scales. These methods constitute the state-of-the-art across a wide range of retrieval and recommendation tasks.

NGAME: Despite these advances, training deep XC models based on transformer encoders poses steep time and memory overheads (see Sec. 3 for a detailed discussion). In particular, popular negative mining techniques often themselves pose memory overheads when used to train large encoder architectures like transformers. This forces mini-batch sizes to remain small leading to slow convergence and sub-optimal accuracies. As a remedy, existing methods either settle for simpler encoders such as bag-of-embeddings [13] or else forego the use of label text altogether in an attempt to speed up training [7, 25, 61], both of which result in sub-optimal accuracies. This paper develops the NGAME method for training large transformer-based Siamese architectures on XC tasks at the scale of 85 million labels. Training is performed in two stages: a pre-training stage first learns an effective Siamese architecture, followed by a fine-tuning stage that freezes the Siamese embeddings and learns a per-label refinement vector to fine-tune the label embeddings to obtain the final classifiers. NGAME is shown to offer 16% more accurate predictions than state-of-the-art methods including BERT-based methods such as LightXML [8], XR-Transformers [61] on a wide array of benchmark datasets in the supervised-learning setting where the label set persists across training and testing. It is notable that this paradigm covers a wide range of real-world applications including related search [23], product recommendation [13, 36, 60], sponsored search [13] and ad recommendation [43].

In live A/B tests on a popular search engine, NGAME was compared on the task of showing personalized ads to users based on their browsing history against an ensemble of state-of-the-art generative, XC, information retrieval (IR) and two-tower models. In these A/B tests, NGAME increased the click-through-rate by 23% over state-of-the-art techniques. On a separate live A/B test on the task of matching user queries to advertiser bid phrases in sponsored search, NGAME increased impressions by 1.3%, clicks by 1.2% and query coverage by 2.1% over leading in-production techniques.

Contributions: The paper makes the following contributions:

1. It notices that existing techniques treat mini-batching and negative mining as unrelated tasks leading to inefficient training. In response, NGAME proposes a light-weight negative mining-aware technique for creating mini-batches. A curious property of the mini-batches so obtained is that in-batch sampling itself starts offering informative negative samples and faster convergence than ANNS-based negative mining techniques such as ANCE (see Section 2). This speeds training up as in-batch sampling has much smaller time and memory overheads than ANNS-based methods.
2. The above technique allows NGAME to operate with much larger mini-batch sizes and allows training with 85 million labels on multiple GPUs without the need of a specialized algorithm to reduce inter-GPU communication. This becomes particularly critical when training large BERT-style encoders which already restrict mini-batch sizes owing to the large memory footprint of their model.
3. Theoretical analysis reveals that NGAME offers provably-accurate negative samples with bounded error under general conditions. Furthermore, under some standard, simplifying assumptions, convergence to a first-order stationary point is also assured.

2 RELATED WORK

Extreme Classification (XC): Early extreme classification algorithms focused primarily on designing accurate and scalable classifiers for either sparse bag-of-words [1–4, 6, 12, 24, 29, 41–45, 54, 56, 59] or dense pre-trained features obtained from feature extraction models such as CDSSM [22, 23] or fastText [26]. More recent work demonstrated that learning task-specific features can lead to significant gains in classification accuracy. The use of diverse feature extraction architectures including Bag-of-Embeddings [14, 17, 18, 36, 62], CNN [10, 30, 32], LSTM [60], and BERT [7, 9, 15, 25, 58] led to the development of *deep extreme classification* (deep XC) techniques that offered significant gains over techniques that either used Bag-of-Words or pre-trained features. However, these techniques did not use label metadata and treated labels as indices devoid of features. Still more recent work [9, 14, 38, 40, 51] used label metadata such as textual descriptions of labels or label correlation graphs. For instance, the X-Transformer [9] and XR-Transformer [61] use label text to cluster labels. However, these methods employ a large ensemble making them expensive to scale. Still more recent techniques have used label metadata in classifier learning more intimately via label text [13, 38], images [39], or label correlation graphs [40, 51]. In all cases, careful use of label metadata is shown to offer better classification accuracies, especially on data-scarce tail labels.

Siamese methods for XC: Although widely studied in areas such as information retrieval [57] and computer vision [20], Siamese methods received attention in XC more recently due to their ability to incorporate label metadata in diverse forms. The key goal in Siamese classification is to learn embeddings of labels and data points such that data points and their positive labels are embedded in close proximity whereas data points and their negative labels maintain a discernible separation. DECAF [38] and ECLARE [40] use asymmetric networks to embed data points and labels whereas SiameseXML [13] and GalaXC [51] use a symmetric network. It is notable that with respect to the type of metadata used, DECAF and SiameseXML use only label text whereas ECLARE and GalaXC use label graphs. DECAF and ECLARE struggle to scale to tasks with over a million labels due to an expensive shortlisting step that seems critical for good performance. On the other hand, GalaXC requires pre-learnt embeddings for data points and labels from models such as Astec [14] that were trained on the same task. This combined with the execution of a multi-layer graph convolutional network makes the method more expensive. Of special interest to this paper is the SiameseXML technique that yields state-of-the-art accuracies on short-text datasets and can scale to 100M labels. However, to achieve such scales, SiameseXML restricts itself to a low-capacity Bag-of-embeddings feature architecture and uses expensive negative sampling techniques that eventually turn out to be suboptimal as discussed below. It is notable that despite the Siamese architecture being well-studied [13, 33, 57], empirical results in this paper demonstrate that existing Siamese training techniques are not optimal with respect to critical steps such as mini-batch selection and negative mining. Thus, extending Siamese architectures to be able to train on 100M-scale datasets with large transformer-based encoders is a non-trivial task requiring careful design of mini-batching and negative sampling routines, as is contributed by this paper.

Negative Mining: Training on a small but carefully selected set of irrelevant labels per training point is critical since training on all irrelevant labels for every data point becomes infeasible when the number of training points and labels are both in the millions. Negative mining techniques come in three flavours:

1. Oblivious: these include random sampling wherein negative samples are either drawn uniformly or from token/unigram distributions [37], and in-batch sampling [11, 13, 16, 17, 20] wherein negative samples are identified among positive samples of other data points within the same mini-batch. The recent works of O-SGD [28] and OWL [49] give theoretical insights into techniques that sample negatives randomly or via in-batch sampling but then either take the hardest among them or else reweigh the sampled negatives according to their hardness. Such methods are computationally inexpensive but experiments suggest that they can offer uninformative negatives and slow convergence [57] (see Appendix B in the supplementary material).

2. Feature-aware: these identify *hard* negatives based on sparse raw features e.g. BM25 [31, 34] or features from a pre-trained model like BERT [21]. However, since these features are not necessarily aligned to the task, the negative samples may offer biased training.

3. Task-aware: AttentionXML [60], XR-Transformer [61], SiameseXML [13] employ a surrogate task [14] to learn label representations and discover negatives using an Approximate Nearest Neighbor Search (ANNS) index over the (now fixed) label representations. LightXML [25] deploys a data structure learnt jointly with the classifiers. However, these techniques are focused on classifier training and can be sub-optimal to train the label embeddings themselves. In contrast, ANCE [57] retrieves hard negatives using task-specific features. As these features keep getting updated during training, the ANNS index is recomputed every few epochs. This offers better negatives but adds substantial time and memory overheads even if the ANNS is updated asynchronously [57] (see Section 3).

In contrast, NGAME provides a strategy for mining provably-accurate negatives similar to task-aware methods (see Theorem 1) but with overheads comparable to those of oblivious techniques. On multiple datasets, the overhead of executing NGAME's negative mining technique was up to 1% as compared to 210% for ANCE (see Tab. 5) allowing NGAME to train with powerful feature architectures such as BERT at extreme scales and offer accuracies superior to leading oblivious, feature- and task-aware negative mining techniques. Due to lack of space, additional details, results and proofs are provided in the supplementary material.

3 NGAME: NEGATIVE MINING-AWARE MINI-BATCHING FOR EXTREME CLASSIFICATION

Notation: L is the total number of labels (the label set remains same during training and testing). $\mathbf{x}_i, \mathbf{z}_l \in \mathcal{X}$ denote textual representations of data point i and label l. The training set $D := \{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}, \{\mathbf{z}_l\}_{l=1}^{L}\}$ has L labels and N data points. For a data point $i \in [N]$, $\mathbf{y}_i \in \{-1, +1\}^L$ is its ground truth vector, *i.e.*, $y_{il} = +1$ if label l is relevant for the data point i and $y_{il} = -1$ otherwise.

Architecture: NGAME uses an encoder model (DistilBERT base [52]) denoted by $\mathcal{E}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{S}^{D-1}$, with trainable parameters $\boldsymbol{\theta}$, to embed both data point and label text onto the D-dimensional unit sphere

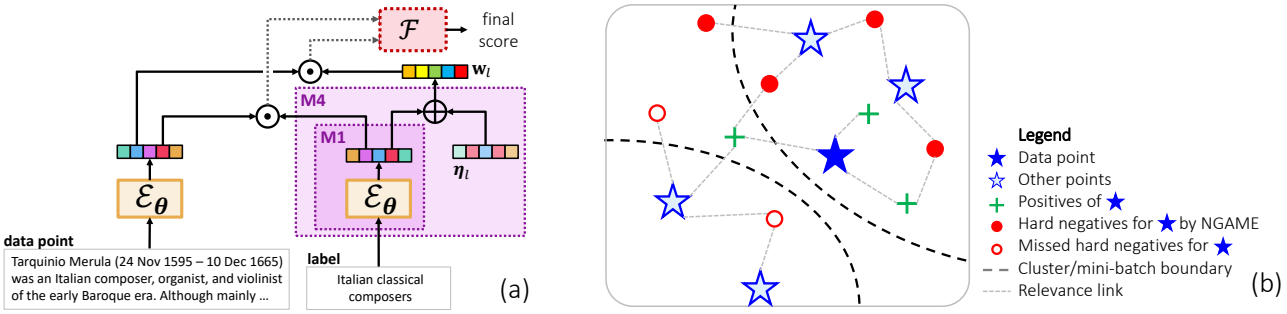

Figure 1: (Left) NGAME's model architecture. Data points and labels are embedded using a Siamese encoder model and per-label 1-vs-all classifiers are learnt. Classifier and Siamese scores are combined to offer final output. (Right) A depiction of NGAME's negative mining strategy. A light dotted line indicates a data point-relevant label pair. NGAME misses only those hard negative labels for a data point that are not relevant to any other data point in its cluster. See also Figure 2 for a real example illustrating the hard-negatives retrieved by NGAME.

$\mathcal{S}^{D-1}$. NGAME also uses 1-vs-all classifiers $\mathbf{W} \overset{\text{def}}{=} \{\mathbf{w}_l\}_{l\in[L]}$ where $\mathbf{w}_l$ is the classifier for label l.

Inference Pipeline: After training $\mathcal{E}_{\boldsymbol{\theta}}, \mathbf{W}$, NGAME establishes a maximum inner product search (MIPS) [35] structure over the set of learnt label classifiers $\{\mathbf{w}_l\}_{l\in[L]}$. Given a test data point $\mathbf{x}_t$, its embedding $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_t)$ is used to invoke the MIPS structure that returns labels $l \in [L]$ with highest classifier scores $\mathbf{w}_l^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_t)$. Mild gains were observed by fusing classifier scores with Siamese scores $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_t)$ for the shortlisted labels using a simple fusion architecture $\mathcal{F}$ and recommending labels in decreasing order of the fused scores (see Appendix C in the supplementary material for details). Appendix D establishes that NGAME offers $O\left(b + D\log L\right)$ time inference where $b = |\boldsymbol{\theta}|$ is the number of parameters in the encoder model.

Joint Training and its Limitations: A triplet loss function could have been used to jointly train $\boldsymbol{\theta}$ and classifiers $\mathbf{W} = \{\mathbf{w}_l\}_{i=1}^{L}$

$$\min_{\boldsymbol{\theta},\mathbf{W}} \mathcal{L}(\boldsymbol{\theta}, \mathbf{W}) = \sum_{i=1}^{N} \sum_{\substack{l:y_{il}=+1 \\ k:y_{ik}=-1}} [\mathbf{w}_k^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{w}_l^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) + \gamma]_+, \quad (1)$$

where the indices l and k run over the relevant and irrelevant labels for data point i respectively with $\mathbf{w}_l, \mathbf{w}_k$ being their label-wise classifiers. Thus, (1) encourages every relevant label to get a score at least margin γ higher than every irrelevant label. Other task losses such as BCE or contrastive are also admissible. However, a training epoch w.r.t. $\mathcal{L}(\boldsymbol{\theta}, \mathbf{W})$ would take $\Omega\left(NL(b + D)\log L\right) = \Omega\left(NL\right)$ time (since data points usually have $O\left(\log L\right)$ relevant labels – see Tab. 1) which is prohibitive when N and L are both in the millions. Joint training also requires storing both the encoder and classifier models in memory. Along with overheads of optimizers such as Adam, this forces mini-batches to be small and slows down convergence [47]. Common workarounds such as distributed training on a GPU cluster [55] impose overheads such as inter-GPU communication. Instead, NGAME proposes a novel two-pronged solution based on modular training and negative mining-aware mini-batching.

Modular Training: NGAME adapts the modular training pipeline outlined in the DeepXML paper [14] that considers four modules (M1 pre-training, M2 negative-mining, M3 feature-transfer, and M4 fine-tuning). To effect this pipeline, NGAME reparameterizes label classifiers as $\mathbf{w}_l = \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) + \boldsymbol{\eta}_l$, where $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$ is the label embedding and $\boldsymbol{\eta}_l$ is a free D-dimensional *residual* vector. This reparameterization is motivated by recent postulates [13, 40] that label embeddings can serve as surrogates for label classifiers. In **module M1**, NGAME sets the residual vectors to zero, *i.e.*, using $\mathbf{w}_l = \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$. Plugging this into (1) results in a loss that depends only on $\boldsymbol{\theta}$ and encourages label-datapoint embedding similarity $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$ to be high when $y_{il} = +1$ and low otherwise. Note that this is identical to Siamese training for encoder models [11, 13, 16, 17, 20, 21, 57] and allows NGAME to perform supervised training of the encoder model $\boldsymbol{\theta}$ in M1. In **module M3**, the learnt encoder model $\hat{\boldsymbol{\theta}}$ is frozen and the classifiers are initialized to $\mathbf{w}_l = \mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{z}_l)$. Finally in **module M4**, the residual vectors $\boldsymbol{\eta}_l$ are implicitly learnt by minimizing (1) with respect to $\mathbf{W}$ alone. It is notable that NGAME does not require a distinct negative mining module and **module M2** (negative mining) is effectively subsumed within M1 and M4 for NGAME. This is possible since NGAME offers negative mining with minimal overhead that can be performed in parallel to training as outlined below. Appendix C in the supplementary material gives further details of M1 and M4 implementation.

Benefits of Modular Training: [14] discusses the benefits of modular training. However, from NGAME's perspective, modular training generalizes the Siamese encoder training strategy popular in literature and makes more effective use of label text during training that benefits rare labels with scant supervision. Splitting encoder and classifier training also eases memory overheads in each module – the parameters and gradients of the encoder and classifier are not required to be stored (in the GPU memory) or computed at the same time. Combined with memory savings offered by NGAME's negative sampling technique (discussed below), this enables NGAME to train with 1.5× larger mini-batches compared to approaches such as ANCE [57] thus offering faster convergence. For instance, NGAME could be trained within 50 hours on 6×V100 GPUs on the PR-85M XC task with more than 85 million labels.

Negative Mining and its Overheads: Modular training partly lowers the memory costs of training but not its $\Omega\left(NL\right)$ computational cost. To reduce computational costs to $O\left(N\log L\right)$, *negative mining* techniques are critical. These restrict training to only the positive/relevant labels for a data point (which are usually only

Algorithm 1 NGAME's negative mining strategy

Require: Init model $\boldsymbol{\theta}^0$, mini-batch size S, cluster size C, refresh interval τ, hardness threshold r

for $t = 0, \ldots, T$ **do**
  if $t\%\tau = 0$ **then** *//Redo clustering at regular intervals*
    Cluster current data point embeddings $\mathcal{E}_{\boldsymbol{\theta}^t}(\mathbf{x}_i)$ into $\lceil N/C \rceil$ clusters, with each cluster containing $\approx C$ data points each.
  end if
  Choose $\lceil S/C \rceil$ random clusters to create a mini-batch S_t of size S
  Take positive labels $\mathcal{P}_+^i$ for each data point $i \in S_t$ and let $L_t \stackrel{\text{def}}{=} \bigcup_{i \in S_t} \mathcal{P}_+^i$
  Compute $\mathcal{E}_{\boldsymbol{\theta}^{t-1}}(\mathbf{x}_i), \mathcal{E}_{\boldsymbol{\theta}^{t-1}}(\mathbf{z}_l)$ for all $i \in S_t, l \in L_t$.
  Select hard-negatives for data point $i \in S_t$ as $\left\{l \in L_t : \|\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)\|_2 \leq r\right\}$
  Update $\boldsymbol{\theta}^t$ using mini-batch SGD over S_t
end for

$O(\log L)$ many – see Tab. 1) and a small set of the hardest-to-classify negative/irrelevant labels for that data point. However, such techniques end up adding their own memory and computational overheads. For example, ANCE [57] queries an ANNS structure over label embeddings that needs to be periodically refreshed (since label embeddings get trained in parallel) while RocketQA [47] trains a cross-encoder to weed out false negatives; these add significant overhead to training. Computing label embeddings $\mathcal{E}_\theta(\mathbf{z}_l)$ for retrieved hard negatives is an additional overhead faced by all these methods. For a mini-batch of, say S training data points, if a single positive and H hard-negatives are chosen per data point, then a total of $S(H+2)$ embeddings need to be computed. For large transformer-based encoder models, this forces batch sizes S to be very small. A notable exception is in-batch negative mining [11, 13, 16, 17, 20, 27] that looks for hard negatives of a data point only among positives of other data points in the same mini-batch and requires computing only $2S$ embeddings. However, since mini-batches are often created randomly, in-batch sampling is known to miss informative negatives and offer slow convergence unless large batches are used [57].

Intuition behind NGAME: The above discussion suggests that inexpensive in-batch sampling technique could have offered good-quality negatives had mini-batch creation encouraged hard-negatives of a data point to exist among positives of other data points in the mini-batch. For example, we note that Siamese training in M1 encourages embeddings of data points and related labels to be close, *i.e.*, $\left\|\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)\right\|_2 \ll 1$ if $y_{il} = 1$. Consequently, hard negative labels are precisely those that are irrelevant yet whose embeddings lie close to that of the data point. Thus, if mini-batches are created out of data points that lie close to each other, triangle inequality would force any in-batch negatives to automatically be good-quality hard ones. This intuition is depicted pictorially in Figure 1 and Theorem 1 assures this formally.

Negative Mining-aware Mini-batching: The resulting mini-batching and negative mining strategy is presented in Algorithm 1 and offers provably accurate hard-negatives at the low computational cost of in-batch negative mining. It also imposes very little compute overhead, requiring occasional data clustering that is inexpensive compared to ANNS creation or cross-encoder training. On a dataset with 85 million labels and 240 million training data points, refreshing ANCE's ANNS index and hard-negative retrieval for all training data points took 4 hours whereas NGAME took under an hour. ANCE needed to compute embeddings for hard negatives separately causing its GPU memory requirement for label embeddings to be at least 2× higher than NGAME that only needed to compute embeddings for positive labels (since NGAME's negatives are sampled from the positives of other data points in the mini-batch). Thus, NGAME imposed a mere 1% increase in epoch time over vanilla in-batch sampling whereas the same was up to 210% for ANCE (see Table 5). NGAME also significantly outperformed techniques such as TAS [21] that use task-agnostic negatives obtained by clustering static features (see Appendix B in the supplementary material). This is because previous works use static one-time clustering [21, 61] using features that are not aligned to the given task and offer poor quality negatives and biased training as a result. On the other hand, NGAME continuously adapts its negatives to the progress made by the training algorithm, thus offering provably accurate negatives and convergent training (see Theorem 1). NGAME also accomplishes this with much smaller overheads than existing state-of-the-art approaches (see Table 5).

Curriculum Learning with NGAME: The use of extremely hard negatives may impact training stability in initial epochs when the model is not well-trained [19]. Thus, it is desirable to initiate training with easier negatives. Fortunately, NGAME can tweak the "hardness" of its negatives by simply changing the cluster size C in Algorithm 1. Using $C = 1$ makes NGAME identical to vanilla in-batch negative mining which offers easier negatives whereas the other extreme $C \to N$ causes NGAME to behave identical to ANNS-based techniques such as ANCE [57] that offer the hardest-to-classify negatives. This allows NGAME to commence training with small values of C and gradually increase it to get progressively harder negatives in later epochs. This curriculum learning approach could lead to 25% – 40% faster convergence when compared to training that used a fixed cluster size C. NGAME further reduces the sampling overhead by 15-20% by using stale embeddings for clustering . NGAME uses a memory-mapped file to store the most fresh embedding for every data-point and uses those to update the clusters instead of computing datapoint embeddings afresh at the time of clustering. Using stale embeddings was not found to significantly affect NGAME's accuracy.

4 THEORETICAL ANALYSIS

Theorem 1 guarantees that negative samples offered by NGAME are provably accurate. The key behind this result is the intuition that NGAME's negative mining strategy should succeed if embeddings of data points and relevant labels lie in close proximity and clustering is sufficiently precise. To state this result, we define the notion of a good embedding and clustering. we note that although the formal statements are presented in terms of Euclidean distances for clarity, these can easily be restated in terms of cosine similarity for encoders that offer normalized embeddings i.e. $\left\|\mathcal{E}_{\boldsymbol{\theta}}(\cdot)\right\|_2 = 1$.

Definition 1 ((r, ϵ)-good embedding). *An encoder model $\mathcal{E}$ parameterized by $\boldsymbol{\theta}$ is said to be (r, ϵ)-good if*

$$\mathop{\mathbb{P}}_{i \in [N], l: y_{il}=1} \left[\left\|\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)\right\|_2 > r\right] \leq \epsilon.$$

Table 1: Dataset Statistics. For all datasets, $L = \Theta(N)$ and data points typically have $O(\log L)$ relevant labels. The public datasets can be downloaded from The Extreme Classification Repository [5].

Dataset	Train N	Labels L	Test N'	Avg. data points per label	Avg. labels per data point
Short-text benchmark datasets					
LF-AmazonTitles-131K	294,805	131,073	134,835	2.29	5.15
LF-AmazonTitles-1.3M	2,248,619	1,305,265	970,237	22.20	38.24
Full-text benchmark datasets					
LF-Amazon-131K	294,805	131,073	134,835	2.29	5.15
LF-WikiSeeAlso-320K	693,082	312,330	177,515	2.11	4.68
LF-Wikipedia-500K	1,813,391	501,070	783,743	4.77	24.75

Definition 2 ((r, ϵ)-good clustering). *A clustering of a given a set of data point embedding vectors $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i)$ into $K = \lceil N/C \rceil$ clusters $C_1, \ldots, C_K$ is said to be (r, ϵ)-good if*

$$\mathop{\mathbb{P}}_{i,j} \left[\left\| \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_j) \right\|_2 \leq r, c(i) \neq c(j) \right] \leq \epsilon,$$

where $c(i) \in [K]$ tells us the cluster to which data point i as assigned.

Definition 3 (r-hard negative). *A label $l \in [L]$ is said to be an r-hard negative label for a datapoint i with respect to an encoder model $\mathcal{E}$ parameterized by $\boldsymbol{\theta}$ if $y_{il} = -1$ and $\left\{ \left\| \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) \right\|_2 \leq r \right\}$*

Theorem 1 (Negative Mining Guarantee). *Suppose Algorithm 1 performs its clustering step using data point embeddings obtained using the encoder model $\mathcal{E}_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ and identifies a set of hard negative labels $\hat{\mathcal{P}}^i_-$ for data point i in step 8 of the algorithm using the threshold $r > 0$. For any $i \in [N], l \in [L]$, let $E^r_{il} := \{y_{il} = -1\} \wedge \left\{ \left\| \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) \right\|_2 \leq r \right\} \wedge \left\{ l \notin \hat{\mathcal{P}}^i_- \right\}$ be the bad event where l is an r-hard negative label for data point i but NGAME fails to retrieve it. Then if the model $\boldsymbol{\theta}$ was (r, ϵ_1)-good and the clustering was $(2r, \epsilon_2)$-good, we are assured that*

$$\frac{1}{NL} \sum_{i \in [N], l \in [L]} \mathbb{I}\left[E^r_{il} \right] \leq c_1 \cdot \epsilon_1 + c_2 \cdot \epsilon_2,$$

for some constants c_1, c_2 that are independent of NGAME's execution and depend only on dataset statistics. For the special case when all data points have the same number of relevant labels and all labels are relevant to the same number of data points, we have $c_1, c_2 \leq 1$.

The special case is described in Corollary 3 in Appendix F.1. NGAME is also able to guarantee convergence to an approximate first-order stationary point (see Theorem 4 in Appendix F.2). For sake of simplicity, this result is presented for Module M1 (Encoder Training) but a similar result holds for Module M4 (Classifier Training) as well.

5 EXPERIMENTS

Datasets: This paper focuses on the supervised learning setting where the label set persists across training and testing phases and considers multiple short- as well as full-text benchmark datasets available at the Extreme Classification Repository [5]. We recall that the supervised-learning setting covers a wide range of applications illustrated in prior works [13, 23, 24, 29, 60, 61]. Both title and detailed content were available for full-text datasets (LF-Amazon-131K, LF-WikiSeeAlso-320K and LF-Wikipedia-500K) whereas only the product/webpage titles were available for the short-text datasets (LF-AmazonTitles-131K and LF-AmazonTitles-1.3M). These datasets cover a variety of applications including product-to-product recommendation (LF-Amazon-131K, LF-AmazonTitles-131K, and LF-AmazonTitles-1.3M), predicting related Wikipedia pages (LF-WikiSeeAlso-320K) and predicting Wikipedia categories (LF-Wikipedia-500K). Please refer to Table 1 for data statistics.

Baselines: Siamese methods for XC such as SiameseXML [13], DECAF [38], and ECLARE [40] are the main baselines for NGAME. Other significant baselines include non-Siamese deep XC methods such as XR-Transformer [61], LightXML [25], BERTXML [7], MACH [36], AttentionXML [60], X-Transformer [9] and Astec [14]. Note that these include methods such as XR-Transformer [61], BERTXML [7] and LightXML [25] that also rely on transformer encoders albeit those that are not trained in a Siamese fashion. For sake of completeness, results are also reported for classical XC methods including Bonsai [29], DiSMEC [2], and Parabel [43] (refer to Appendix A in the supplementary material for all results). Implementations provided by the respective authors were used for all methods.

Hyper-parameters: NGAME's hyper-parameters are: (i) cluster size, (ii) batch size, (iii) interval τ between clustering updates and (iv) margin value γ. The Adam optimizer was used to learn model parameters and its hyper-parameters include learning rate and number of epochs. NGAME did not require much hyper-parameter tuning and default values were used for all hyper-parameters except for number of epochs. Please see Appendix C in supplementary material for a detailed discussion on hyper-parameters. NGAME's encoder $\mathcal{E}_{\boldsymbol{\theta}}$ was initialized with the 6-layered DistilBERT base [52] to encode data points and labels. The hyper-parameters of baseline algorithms were set as suggested by their authors wherever applicable and by fine-grained validation otherwise.

Evaluation metrics: Algorithms were evaluated using popular metrics such as precision@k (P@k, $k \in 1, 5$) and their propensity-scored [24, 46] variants precision@k (PSP@k, $k \in 1, 5$). Results on other metrics such as nDCG@k (N@k) & propensity scored nDCG@k (PSN@k) are included in Appendix E in the supplementary material. It is notable that the propensity-scored metrics are known to measure performance on tail labels [24]. Definitions of all these metrics are available at [5] and definitions of metrics used in A/B testing experiments are also provided in Appendix E.

Offline evaluation on benchmark XC datasets: NGAME's P@1 could be up to 16% higher than leading Siamese methods for XC including SiameseXML, ECLARE, and DECAF which are the focus of the paper. This demonstrates that NGAME's design choices lead to significant gains over these methods. Please refer to the ablation experiments in Appendix B for detailed discussion on impact of NGAME's components.

Table 2 presents results on full-text datasets where NGAME could also be upto 13% and 15% more accurate in P@k and PSP@k respectively when compared to leading deepXC methods such as LightXML and XR-Transformer. It is notable that these methods also use transformer architectures indicating the effectiveness of NGAME's training pipeline. It is also notable that NGAME outperforms a range of other negative sampling algorithms such as TAS [21], DPR [27] and ANCE [61]. Similar results were observed on short-text datasets, where NGAME could be up to 11% and 13% more

Table 2: Results on full-text benchmark datasets. TT refers to training time in hours on a single Nvidia V100 GPU. See the supplementary material for full results and detailed discussion on accuracies and training time.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	TT
LF-Wikipedia-500K							
NGAME	**84.01**	**64.69**	**49.97**	**41.25**	**52.57**	**57.04**	54.88
SiameseXML	67.26	44.82	33.73	33.95	35.46	37.07	4.37
ECLARE	68.04	46.44	35.74	31.02	35.39	38.29	9.4
DECAF	73.96	54.17	42.43	32.13	40.13	44.59	13.4
XR-Transformer	81.62	61.38	47.85	33.58	42.97	47.81	119.47
LightXML	81.59	61.78	47.64	31.99	42	46.53	249
Astec	73.02	52.02	40.53	30.69	36.48	40.38	6.39
Bonsai	69.2	49.8	38.8	27.46	32.25	35.48	1.39
LF-WikiSeeAlso-320K							
NGAME	**47.65**	**31.56**	**23.68**	**33.83**	**37.79**	**41.03**	75.39
SiamseXML	42.16	28.14	21.39	29.02	2.68	36.03	2.33
ECLARE	40.58	26.86	20.14	26.04	30.09	33.01	9.4
DECAF	41.36	28.04	21.38	25.72	30.93	34.89	13.4
XR-Transformers	42.57	28.24	21.3	25.18	30.13	33.79	119.47
LightXML	34.5	22.31	16.83	17.85	21.26	24.16	249
BERTXML	42.63	27.65	20.41	26.16	31.41	34.63	116.67
Astec	40.07	26.69	20.36	23.41	28.08	31.92	6.39
Bonsai	34.86	23.21	17.66	18.19	22.35	25.66	1.39
MACH	34.52	23.39	17	25.27	30.71	35.42	13.91
LF-Amazon-131K							
NGAME	**46.53**	**30.89**	**22.02**	**38.53**	**44.95**	**50.45**	39.99
SiameseXML	44.81	30.19	21.94	37.56	43.69	49.75	1.18
ECLARE	43.56	29.65	21.57	34.98	42.38	48.53	2.15
DECAF	42.94	28.79	21	34.52	41.14	47.33	1.8
XR-Transformer	45.61	**30.85**	22.32	34.93	42.83	49.24	38.4
LightXML	41.49	28.32	20.75	30.27	37.71	44.1	56.03
BERTXML	42.59	28.39	20.27	33.55	40.83	46.4	48.11
Astec	42.22	28.62	20.85	32.95	39.42	45.3	3.05
Bonsai	40.23	27.29	19.87	29.6	36.52	42.39	0.4
MACH	34.52	23.39	17	25.27	30.71	35.42	13.91

Table 3: Results on short-text benchmark datasets. See the supplementary material for full results. TT refers to training time in hours on a single Nvidia V100 GPU. – denotes that results are unavailable.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	TT
LF-AmazonTitles-1.3M							
NGAME	**56.75**	**49.19**	**44.09**	**29.18**	**33.01**	**35.36**	97.75
SiameseXML	49.02	42.72	38.52	27.12	30.43	32.52	9.89
ECLARE	50.14	44.09	40	23.43	27.9	30.56	70.59
DECAF	50.67	44.49	40.35	22.07	26.54	29.3	74.47
XR-Transformer	50.14	44.07	39.98	20.06	24.85	27.79	132
LightXML	–	–	–	–	–	–	–
Astec	48.82	42.62	38.44	21.47	25.41	27.86	18.54
Bonsai	47.87	42.19	38.34	18.48	23.06	25.95	7.89
LF-AmazonTitles-131K							
NGAME	**46.01**	**30.28**	**21.47**	**38.81**	**44.4**	**49.43**	12.59
SiameseXML	41.42	27.92	21.21	35.8	40.96	46.19	1.08
ECLARE	40.74	27.54	19.88	33.51	39.55	44.7	2.16
DECAF	38.4	25.84	18.65	30.85	36.44	41.42	2.16
XR-Transformer	38.1	25.57	18.32	28.86	34.85	39.59	35.4
LightXML	35.6	24.15	17.45	25.67	31.66	36.44	71.4
BERTXML	38.89	26.17	18.72	30.1	36.81	41.85	12.55
Astec	37.12	25.2	18.24	29.22	34.64	39.49	1.83
Bonsai	34.11	23.06	16.63	24.75	30.35	34.86	0.1

Table 4: Results on the PR-85M dataset for personalized ad recommendations

Method	P@1	P@3	P@5	N@5	R@5	PSP@1	PSP@5
NGAME	**30.77**	**18.09**	**13.20**	**32.46**	**33.82**	24.90	32.87
ANCE	24.97	15.52	11.72	28.68	31.64	**26.48**	**33.55**
SiameseXML	27.60	16.45	12.59	30.28	30.61	17.46	24.89

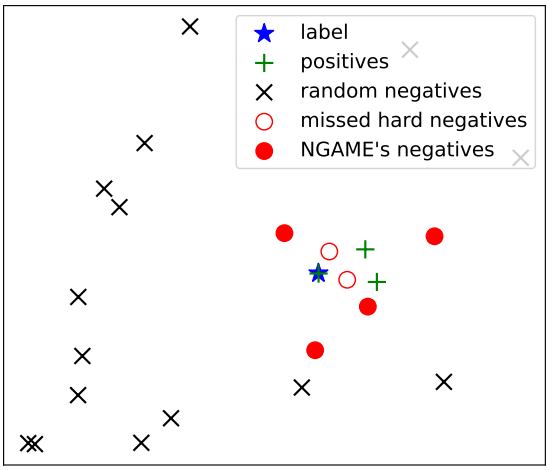

Figure 2: t-SNE representation of positives, random-negatives, NGAME's negatives and hard negatives missed by NGAME for a product titled *'Fearless Confessions: A Writer's Guide to Memoir'*. Note that NGAME recovers most of the hard negatives for the data point.

accurate in P@1 and PSP@1 respectively, compared to specialized algorithms designed for short-texts including SiameseXML, Astec, DECAF, *etc.* NGAME also continues to outperform other negative sampling algorithms such as TAS [21], DPR [27], and ANCE [61].

Curiously, [14] observed that jointly training a high-capacity feature extractor such as BERT along with the classification layer (as opposed to training them in a modular manner as done by NGAME) could yield inferior results, especially on short-text datasets. We observe a similar trend where NGAME's pipeline yielded 7% better P@1 as compared to the same architecture trained in an end-to-end manner (referred as BERTXML in Table 3). The multi-stage training employed by NGAME where the transformer-based encoder is first trained in a Siamese fashion was found to address this challenge and yield state-of-the-art results on short-text datasets as well.

Analysis of Results: Tail labels contribute significantly to the performance of both components of NGAME's classifier – $\mathcal{E}_{\theta}(\mathbf{z}_l)$ and $\mathbf{w}_l$. Appendix A presents decile-wise analysis indicating that NGAME derives its superior performance not just by predicting popular labels but predicting rare labels accurately as well. The same was corroborated by NGAME's performance in live A/B testing on Sponsored Search where it was able to predict labels that were not being predicted by the existing ensemble of methods. NGAME's final predictions which make use of both components get the best out of both leading to overall superior performance.

Table 5: Breakdown of computation costs of different negative mining methods. Fraction increase refers to the ratio of a method's total time compared to DPR. The indexing and querying times of ANNS-based algorithms such as ANCE depend on dataset statistics causing the overheads to vary.

Metric	DPR	NGAME	ANCE	DPR	NGAME	ANCE
	LF-Wikipedia-500K			LF-AmazonTitles-1.3M		
Epoch time	4710s	4710s	6360s	1092s	1092s	3120s
Sampling overhead	-	6s	1131.22s	-	12s	262.16s
Total time	4710s	4716s	7491.22s	1092s	1104s	3382.16s
Fraction Increase	-	1.00	1.59	-	1.01	3.10

Live A/B testing and offline evaluation on Personalized Ad Recommendation: NGAME was used to predict queries that could lead to clicks on a given webpage in the pipeline to show personalized ads to users and was compared to an existing ensemble of state-of-the-art IR, dense retrieval (DR) and XC techniques. In A/B tests on live traffic, NGAME was found to increase Click-Through Rate (CTR) and Click Yield (CY) by 23% and 19% respectively. In manual labeling by expert judges, NGAME was found to increase the quality of predictions, measured in terms of fraction of excellent and good predictions, by 16% over the ensemble of baselines. The PR-85M dataset was created to capture this inverted prediction task by mining the search engine logs for a specific time period where each webpage title became a data point and search engine queries that led to a click on that webpage became labels relevant to that data point. NGAME was found to be at least 2% more accurate than leading XC as well as Siamese encoder-based methods including ANCE and SiameseXML in R@5 metric on the PR-85M dataset. Please see Table 4 for detailed results.

Live A/B testing on Sponsored Search: NGAME was deployed on the Bing search engine and A/B tests were performed on live search engine traffic for matching user queries to advertiser bid phrases (Query2Bid). NGAME was compared to an ensemble of leading (proprietary) IR, XC, generative and graph-based techniques. NGAME was found to increase Impression Yield (IY), Click Yield (CY) and Query Coverage (QC) by 1.3%, 1.23% and 2.12%, respectively. The IY boost indicates that NGAME discovered more ads not being captured by existing algorithms. The CY boost indicates that ads surfaced by NGAME were more relevant to the end user. The QC boost indicates that NGAME impressed ads on several queries for which ads were previously not being shown.

6 ABLATIONS

To assess the impact of NGAME's negative mining technique, ablation experiments instead used popular negative mining algorithms *e.g.,* TAS [21], DPR [27], O-SGD [28], and ANCE [61]. Code was unavailable for OWL [49]. The pipelines trained using NGAME yielded 1.8-7% gains in accuracy over pipelines trained using DPR, TAS, and ANCE (Please see Table 6 for a detailed comparison). Note that NGAME provides task-aware negatives whereas negatives offered by DPR and TAS negatives are not updated as training progresses. Additionally, NGAME's semi-hard negatives offered more stable and accurate training [19, 53] than ANCE's globally hard negatives especially on XC tasks where missing labels abound [13, 24, 46].

Table 6: Ablation experiments for different negative mining strategies. TT refers to training time in hours on a single Nvidia V100 GPU.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	TT
	LF-Wikipedia-500K						
NGAME	**84.01**	**64.69**	**49.97**	**41.25**	**52.57**	**57.04**	54.88
DPR	79.91	59.51	45.9	37.57	46.51	50.7	54.67
TAS	82.23	62.7	48.36	38.43	48.38	52.83	54.68
ANCE	76.9	57.64	45.1	37.75	44.65	48.85	75.08
	LF-AmazonTitles-1.3M						
NGAME	**56.75**	**49.19**	**44.09**	29.18	33.01	35.36	97.75
DPR	51.87	45.85	41.34	29.93	34.49	37.08	96.83
TAS	51.2	44.65	40	28.53	32.03	34	96.87
ANCE	53.32	46.61	40.24	**31.47**	**34.97**	**35.67**	447.25

Table 5 presents the sampling overhead of different algorithms – NGAME's overhead was just 1% as compared to 210% for ANCE. The HNSW algorithm [35] was used for nearest neighbor search for ANCE whose indexing and querying time depends on the number of labels. Moreover, NGAME offers superior accuracies as well as faster convergence compare to a range of existing negative mining algorithms such as TAS, DPR, O-SGD, and ANCE (see Appendix B). RocketQA [47] could only scale to small datasets, *i.e.,* LF-AmazonTitles-131K where it led to a marginal gain of 0.5% over NGAME. Please see Appendix C for implementation details.

Appendix B in the supplementary material considers different variants of NGAME's prediction pipeline such as one that makes predictions only using label embeddings ($\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$), one that uses only label classifiers ($\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_l$), and one using NGAME's score fusion function ($\mathcal{F}(\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l), \mathbf{w}_l, \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))$). BERT-Base [15] led to marginal gains over DistilBERT-Base [50] but at the cost of 1.8× increase in training time (See Appendix B in the supplementary material). This establishes that NGAME's gains are not attributable to its encoder alone and that NGAME improves performance for a wide range of encoder architectures.

7 CONCLUSIONS AND FUTURE WORK

This work accelerates the training of XC architectures that use large Siamese encoders such as transformers. A key step towards doing this is the identification of negative mining techniques as a bottleneck that forces mini-batch sizes to remain small and in turn, slows down convergence. The paper proposes the NGAME method that uses negative-mining-aware mini-batch creation to train Siamese XC methods and effectively train large encoder architectures such as transformers making effective use of label-text. There exist other forms of label-metadata that have been exploited in XC literature, for instance, label hierarchies and correlation graphs. Although experiments show that NGAME outperforms these methods by use of more powerful architectures alone, it remains to be seen how NGAME variants using graph/tree metadata would perform. In terms of theoretical results, it is a tantalizing opportunity to relate the (ϵ, r)-goodness of the embedding model to the training loss of the model. This would set up a virtuous cycle and possibly offer stronger convergence proofs.

REFERENCES

[1] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*.

[2] R. Babbar and B. Schölkopf. 2017. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *WSDM*.

[3] R. Babbar and B. Schölkopf. 2019. Data scarcity, robustness and extreme multi-label classification. *ML* (2019).

[4] E. J. Barezi, I. D. W., P. Fung, and H. R. Rabiee. 2019. A Submodular Feature-Aware Framework for Label Subset Selection in Extreme Classification Problems. In *NAACL*.

[5] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The Extreme Classification Repository: Multi-label Datasets & Code. http://manikvarma.org/downloads/XC/XMLRepository.html

[6] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In *NIPS*.

[7] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. 2019. Extreme Multi-Label Legal Text Classification: A case study in EU Legislation. In *ACL*.

[8] C. W. Chang, H. F. Yu, K. Zhong, Y. Yang, and I. S. Dhillon. 2019. A Modular Deep Learning Approach for Extreme Multi-label Text Classification. *CoRR* (2019).

[9] W. C. Chang, Yu H. F., K. Zhong, Y. Yang, and I. S. Dhillon. 2020. Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *KDD*.

[10] S.-A. Chen, J.-J. Liu, T.-H. Yang, H.-T. Lin, and C.-J. Lin. 2022. Even the Simplest Baseline Needs Careful Re-investigation: A Case Study on XML-CNN. In *NAACL*.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

[12] M Cissé, N. Usunier, T. Artières, and P. Gallinari. 2013. Robust bloom filters for large multilabel classification tasks. In *NIPS*.

[13] K. Dahiya, A. Agarwal, D. Saini, K. Gururaj, J. Jiao, A. Singh, S. Agarwal, P. Kar, and M. Varma. 2021. SiameseXML: Siamese Networks meet Extreme Classifiers with 100M Labels. In *ICML*.

[14] K. Dahiya, D. Saini, A. Mittal, A. Shaw, K. Dave, A. Soni, H. Jain, S. Agarwal, and M. Varma. 2021. DeepXML: A Deep Extreme Multi-Label Learning Framework Applied to Short Text Documents. In *WSDM*.

[15] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL* (2019).

[16] F. Faghri, D.-J. Fleet, J.-R. Kiros, and S. Fidler. 2018. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. In *BMVC*.

[17] C. Guo, A. Mousavi, X. Wu, D.-N. Holtmann-Rice, S. Kale, S. Reddi, and S. Kumar. 2019. Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces. In *NeurIPS*.

[18] V. Gupta, R. Wadbude, N. Natarajan, H. Karnick, P. Jain, and P. Rai. 2019. Distributional Semantics Meets Multi-Label Learning. In *AAAI*.

[19] B. Harwood, Kumar B.-V., G. Carneiro, I. Reid, and T. Drummond. 2017. Smart mining for deep metric learning. In *ICCV*.

[20] K. He, Haoqi Fan, Yuxin W., S. Xie, and R. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

[21] S. Hofstätter, S.-C. Lin, J.-H. Yang, J. Lin, and A. Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *SIGIR*.

[22] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *CIKM*.

[23] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. 2019. Slice: Scalable Linear Extreme Classifiers trained on 100 Million Labels for Related Searches. In *WSDM*.

[24] H. Jain, Y. Prabhu, and M. Varma. 2016. Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking and Other Missing Label Applications. In *KDD*.

[25] T. Jiang, D. Wang, L. Sun, H. Yang, Z. Zhao, and F. Zhuang. 2021. LightXML: Transformer with Dynamic Negative Sampling for High-Performance Extreme Multi-label Text Classification. In *AAAI*.

[26] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *EACL*.

[27] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-T. Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*.

[28] K. Kawaguchi and H. Lu. 2020. Ordered SGD: A New Stochastic Optimization Framework for Empirical Risk Minimization. In *AISTATS*.

[29] S. Khandagale, H. Xiao, and R. Babbar. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *ML* (2020).

[30] S. Kharbanda, A. Banerjee, A. Palrecha, and R. Babbar. 2021. Embedding Convolutions for Short Text Extreme Classification with Millions of Labels. *arXiv preprint arXiv:2109.07319* (2021).

[31] M. C. Lee, B. Gao, and R. Zhang. 2018. Rare Query Expansion Through Generative Adversarial Networks in Search Advertising. In *KDD*.

[32] J. Liu, W. Chang, Y. Wu, and Y. Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*.

[33] W. Lu, J. Jiao, and R. Zhang. 2020. TwinBERT: Distilling Knowledge to Twin-Structured Compressed BERT Models for Large-Scale Retrieval. In *CIKM*.

[34] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins. 2020. Sparse, Dense, and Attentional Representations for Text Retrieval. *TACL*.

[35] A. Y. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *TPAMI* (2020).

[36] T. K. R. Medini, Q. Huang, Y. Wang, V. Mohan, and A. Shrivastava. 2019. Extreme Classification in Log Memory using Count-Min Sketch: A Case Study of Amazon Search with 50M Products. In *NeurIPS*.

[37] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*.

[38] A. Mittal, K. Dahiya, S. Agrawal, D. Saini, S. Agarwal, P. Kar, and M. Varma. 2021. DECAF: Deep Extreme Classification with Label Features. In *WSDM*.

[39] A. Mittal, K. Dahiya, S. Malani, J. Ramaswamy, S. Kuruvilla, J. Ajmera, K. Chang, S. Agrawal, P. Kar, and M. Varma. 2022. Multimodal Extreme Classification. In *CVPR*.

[40] A. Mittal, N. Sachdeva, S. Agrawal, S. Agarwal, P. Kar, and M. Varma. 2021. ECLARE: Extreme Classification with Label Graph Correlations. In *WWW*.

[41] R. Panda, A. Pensia, N. Mehta, M. Zhou, and P. Rai. 2019. Deep Topic Models for Multi-label Learning. In *ICML*.

[42] Y. Prabhu, A. Kag, S. Gopinath, K. Dahiya, S. Harsola, R. Agrawal, and M. Varma. 2018. Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation. In *WSDM*.

[43] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*.

[44] Y. Prabhu, A. Kusupati, N. Gupta, and M. Varma. 2020. Extreme Regression for Dynamic Search Advertising. In *WSDM*.

[45] Y. Prabhu and M. Varma. 2014. FastXML: A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning. In *KDD*.

[46] M. Qaraei, E. Schultheis, P. Gupta, and R. Babbar. 2021. Convex Surrogates for Unbiased Loss Functions in Extreme Classification With Missing Labels. In *The WebConf*.

[47] Y. Qu, Y. Ding, J. Liu, K. Liu, R. Ren, W. X. Zhao, D. Dong, H. Wu, and H. Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering.

[48] A. S. Rawat, A. K. Menon, W. Jitkrittum, S. Jayasumana, F. X. Yu, S. Reddi, and S. Kumar. 2021. Disentangling Sampling and Labeling Bias for Learning in Large-Output Spaces. In *ICML*.

[49] S. J. Reddi, S. Kale, F.X. Yu, D. N. H. Rice, J. Chen, and S. Kumar. 2019. Stochastic Negative Mining for Learning with Large Output Spaces. In *AISTATS*.

[50] N. Reimers and I. Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *EMNLP* (2019).

[51] D. Saini, A.K. Jain, K. Dave, J. Jiao, A. Singh, R. Zhang, and M. Varma. 2021. GalaXC: Graph Neural Networks with Labelwise Attention for Extreme Classification. In *WWW*.

[52] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* (2019).

[53] F. Schroff, D. Kalenichenko, and J. Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.

[54] W. Siblini, P. Kuntz, and F. Meyer. 2018. CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. In *ICML*.

[55] L. Song, P. Pan, K. Zhao, H. Yang, Y. Chen, Y. Zhang, Y. Xu, and R. Jin. 2020. Large-Scale Training System for 100-Million Classification at Alibaba. In *KDD*.

[56] T. Wei, W. W. Tu, and Y. F. Li. 2019. Learning for Tail Label Data: A Label-Specific Feature Approach. In *IJCAI*.

[57] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.

[58] H. Ye, Z. Chen, D.-H. Wang, and B. D. Davison. 2020. Pretrained Generalized Autoregressive Model with Adaptive Probabilistic Label Clusters for Extreme Multi-label Text Classification. In *ICML*.

[59] E. H. I. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing. 2017. PPDSparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *KDD*.

[60] R. You, S. Dai, Z. Zhang, H. Mamitsuka, and S. Zhu. 2019. AttentionXML: Extreme Multi-Label Text Classification with Multi-Label Attention Based Recurrent Neural Networks. In *NeurIPS*.

[61] J. Zhang, W. C. Chang, H. F. Yu, and I. Dhillon. 2021. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In *NeurIPS*.

[62] W. Zhang, L. Wang, J. Yan, X. Wang, and H. Zha. 2018. Deep Extreme Multi-label Learning. *ICMR* (2018).