# SiameseXML: Siamese Networks meet Extreme Classifiers with 100M Labels

**Kunal Dahiya** [1] **Ananye Agarwal** [1] **Deepak Saini** [2] **Gururaj K** [3] **Jian Jiao** [3] **Amit Singh** [3] **Sumeet Agarwal** [1]
**Purushottam Kar** [4 2] **Manik Varma** [2 1]

## Abstract

Deep extreme multi-label learning (XML) requires training deep architectures that can tag a data point with its most relevant subset of labels from an extremely large label set. XML applications such as ad and product recommendation involve labels rarely seen during training but which nevertheless hold the key to recommendations that delight users. Effective utilization of label metadata and high quality predictions for rare labels at the scale of millions of labels are thus key challenges in contemporary XML research. To address these, this paper develops the SiameseXML framework based on a novel probabilistic model that naturally motivates a modular approach melding Siamese architectures with high-capacity extreme classifiers, and a training pipeline that effortlessly scales to tasks with 100 million labels. SiameseXML offers predictions 2–13% more accurate than leading XML methods on public benchmark datasets, as well as in live A/B tests on the Bing search engine, it offers significant gains in click-through-rates, coverage, revenue and other online metrics over state-of-the-art techniques currently in production. Code for SiameseXML is available at https://github.com/Extreme-classification/siamesexml

## 1. Introduction

**Overview**: Extreme Multi-label Learning (XML) involves tagging a data point with its most relevant subset of labels from an extremely large set. XML finds applications in myriad of ranking and recommendation tasks such as product-to-product (Mittal et al., 2021a), product-to-query (Chang et al., 2020a), query-to-product (Medini et al., 2019), query-to-bid-phrase (Dahiya et al., 2021), etc. Applications where data points are endowed with short (3-10 tokens) textual descriptions e.g., product title, search query are known as *short-text* applications. These accurately model the ranking and recommendation tasks mentioned above and have attracted much attention recently (Chang et al., 2020a; Dahiya et al., 2021; Medini et al., 2019; Mittal et al., 2021a).

**XML with Label Metadata**: Recent works such as (Mittal et al., 2021a) have demonstrated that for applications where labels (e.g., related items, bid-phrases) are also endowed with textual descriptions, the use of such label metadata is beneficial in accurately predicting *rare* labels for which very little training data (often $< 5$ training points) is available and hence, training data alone may not inform the classifier model adequately. This is in sharp contrast with XML works e.g., (Babbar & Schölkopf, 2019; Prabhu et al., 2018b; You et al., 2019) that treat labels as identifiers devoid of descriptive features. Other forms of label metadata such as label hierarchies could also be used but this paper focuses on label text as a readily available form of label metadata.

**Beyond Siamese Architectures**: The presence of textual descriptions on both data-point and label sides invites the application of *Siamese* models (Chen et al., 2020; Schroff et al., 2015; Yeh et al., 2017; Xiong et al., 2020) that use a shared architecture to embed both data-points and labels such that related data-points and labels are embedded in close proximity. However existing works on Siamese models focus largely on *zero-shot* cases i.e. predicting hitherto unseen labels. While zero-shot cases present an important use-case in several applications, an equally critical use-case in ranking and recommendation scenarios is that of *few-shot* labels which are indeed seen during training but perhaps rarely. Note that the prediction problem for few-shot labels can be turned into a *classification* problem rather than a mere retrieval problem. This presents opportunities for models beyond those that are *purely* Siamese.

**Contributions**: This paper presents the SiameseXML framework that generalizes existing Siamese models in few-shot scenarios by melding Siamese architectures with per-label extreme classifiers at the scale of 100 million labels. The SiameseXML framework is **a)** based on a novel probabilistic model that naturally motivates a modular approach melding Siamese networks with extreme classifiers and **b)** offers generalization bounds for its Siamese module that are independent of the number of labels $L$ based on a novel

[1]Indian Institute of Technology Delhi [2]Microsoft Research [3]Microsoft [4]Indian Institute of Technology Kanpur. Correspondence to: Kunal Dahiya <kunalsdahiya@gmail.com>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. 

Maurey-type sparsification lemma that may be of independent interest. An implementation of SiameseXML could **c)** scale to tasks with 100M labels while still offering predictions within milliseconds and **d)** offer predictions 2-13% more accurate than leading XML methods on benchmark datasets as well as improve prediction quality by 11% when matching user queries to advertiser bid phrases.

## 2. Related Works

**Extreme Multi-label Learning (XML)**: Much prior work has focused on designing classifiers for fixed features such as bag-of-words or else pre-trained features such as FastText (Joulin et al., 2017). Representative works include (Agrawal et al., 2013; Babbar & Schölkopf, 2017; 2019; Bhatia et al., 2015; Jain et al., 2016; 2019; Jasinska et al., 2016; Khandagale et al., 2019; Mineiro & Karampatziakis, 2015; Niculescu-Mizil & Abbasnejad, 2017; Prabhu & Varma, 2014; Prabhu et al., 2018a;b; Tagami, 2017; Xu et al., 2016; Yen et al., 2016; Prabhu et al., 2020). However, recent works offer superior accuracy by using deep learning algorithms to jointly learn task-dependent features and classifiers, even at the scale of millions of labels. These include XML-CNN (Liu et al., 2017), AttentionXML (You et al., 2019), MACH (Medini et al., 2019), Astec (Dahiya et al., 2021), X-Transformer (Chang et al., 2020a), XT (Wydmuch et al., 2018), APLC-XLNet (Ye et al., 2020), and LightXML (Jiang et al., 2021) that respectively use CNN, attention, MLP, bag-of-embeddings, and transformer-based architectures.

**Label Metadata**: Recent works have demonstrated that incorporating label metadata can significantly improve accuracy even compared to deep learning approaches that consider labels as feature-less identifiers e.g. Astec (Dahiya et al., 2021). Prominent among them are the X-Transformer (Chang et al., 2020a) and DECAF (Mittal et al., 2021a) that make use of label text and ECLARE (Mittal et al., 2021b) and GalaXC (Saini et al., 2021) that make use of label graphs. SiameseXML empirically outperforms all these methods on a variety of benchmark datasets.

The X-Transformer makes use of label text to learn intermediate representations and learns a vast ensemble of classifiers based on the powerful transformer architecture. However, training multiple transformer models requires a rather large array of GPUs and the method has not been shown to scale to several millions of labels. On the other hand, DECAF uses label text embeddings as a component in its 1-vs-all label classifiers. Although relatively more scalable, DECAF still struggles with several millions of labels. The method crucially relies on clustering labels into $\hat{L}$ *meta-labels* and seems to demand progressively larger and larger values of $\hat{L}$ as the number of labels $L$ grows, for example $\hat{L} \approx 130$K for $L \approx 1$M labels. Since DECAF requires meta-label representations corresponding to all $\hat{L}$ meta labels to be recomputed for every mini-batch, it makes the method expensive for larger datasets. Taking a small value, say $\hat{L} \approx 8$K improves speed but hurts performance. SiameseXML instead uses direct label-text embeddings to offer better scalability and accuracy than DECAF's meta-label based approach.

**Siamese Networks**: Siamese networks typically learn data point and label embeddings by optimizing the pairwise contrastive loss (Chen et al., 2020; Xiong et al., 2020) or the triplet loss (Schroff et al., 2015; Wu et al., 2017) on a variety of pre-training tasks such as Cloze task and related pair prediction (Chang et al., 2020b). However, these approaches focus mainly on zero-shot labels and neglect opportunities presented by few-shot labels. SiameseXML shows that for few-shot scenarios where labels are not entirely unseen but maybe just rarely seen, Siamese architectures can be significantly empowered using extreme classifiers.

**Negative Mining**: Performing optimizations on Siamese models at extreme scales can be computationally prohibitive as for $N$ data points and $L$ labels, a per-epoch cost of respectively $\mathcal{O}(NL)$ and $\mathcal{O}\left(NL^2\right)$ is incurred for pairwise and triplet losses. To avoid this, it is common to train a data point with respect to only a small, say $\mathcal{O}(\log L)$-sized, subset of labels which brings training cost down to $\mathcal{O}(N \log L)$ per epoch. This subset typically contains all positive labels of the data point (of which there are typically $\mathcal{O}(\log L)$ in several applications with large $L$) and a carefully chosen set of $\mathcal{O}(\log L)$ negative labels which seem the most challenging for this data point. There is some debate as to whether the absolutely "hardest" negatives for a data point should be considered or not (Wu et al., 2017; Xiong et al., 2020), especially in situations where missing labels abound (Jain et al., 2016), or whether considering multiple hard negatives is essential. For instance, (Schroff et al., 2015; Harwood et al., 2017) observed that using only the "hardest" of negatives could lead to bad local minima or collapsed models. Nevertheless, such *negative mining* has become a cornerstone for training classifier models at extreme scales. Although relatively well-understood for approaches that use fixed features e.g. (Jain et al., 2019), negative mining becomes an interesting problem in itself when features get jointly learnt since the set of challenging negatives for a data point may keep changing as the feature representation of that data point changes across epochs. Several approaches have been proposed to address this. Online *in-batch* approaches look for challenging negative labels for a data point within the positive labels of other data points within the mini-batch (Faghri et al., 2018; Guo et al., 2019; Chen et al., 2020; He et al., 2020). Although computationally cheap, as the number of labels grows, it becomes more and more unlikely that the most useful negative labels for a data point would just happen to get sampled within its mini-batch. Thus, at extreme scales, either negative mining quality suf-

fers or else mini-batch sizes have to be enlarged which hurts scalability (Chen et al., 2020; Dahiya et al., 2021). Offline approaches that utilize approximate nearest neighbor search (ANNS) structures have also been studied. This can either be done once if using pre-trained features (Luan et al., 2020) or else periodically if using learnt features (Xiong et al., 2020; Harwood et al., 2017). However, repeated creation of ANNS structures can be expensive in itself.

## 3. The SiameseXML Framework

This section develops the SiameseXML framework based on a novel probabilistic model that naturally motivates a modular architecture melding Siamese models with extreme classifiers. The framework is developed as an extension of the DeepXML framework (Dahiya et al., 2021).

**Notation**: Let $\{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, \{\mathbf{z}_l\}_{l=1}^L\}$ be a multi-label training set with $L$ labels and $N$ data points. $\mathbf{x}_i, \mathbf{z}_l \in \mathcal{X}$ denote representations of the textual descriptions of data point $i$ and label $l$ respectively. For each $i \in [N]$, $\mathbf{y}_i \in \{-1, +1\}^L$ denotes the ground truth label vector for data point $i$ such that $y_{il} = +1$ if label $l$ is relevant to data point $i$ else $y_{il} = -1$. $\mathcal{S}^{D-1}$ denotes the $D$-dimensional unit sphere.

**Probabilistic Model**: Let $\mathcal{E}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{S}^{D-1}$ denote a parameterized embedding model with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. The first postulate below posits that "extreme" 1-vs-all (OvA) classifiers suffice to solve the XML problem. This postulate is widely accepted by 1-vs-all methods popular in literature, e.g., (Babbar & Schölkopf, 2019; Chang et al., 2020a; Jain et al., 2019; Prabhu et al., 2018b; Yen et al., 2017).

**Postulate 1** (OvA sufficiency)**.** *For some known (joint) likelihood function $P_L : \{-1, +1\}^L \times [-1, 1]^L \to [0, 1]$, unknown parameters $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}$, and per-label extreme classifiers i.e. $\mathbf{w}_l^* \in S^{D-1}$ for all $l \in [L]$, we have*

$$\mathbb{P}\left[\mathbf{y} \,|\, \mathbf{x}_i, \{\mathbf{w}_l^*\}, \boldsymbol{\theta}^*\right] = P_L\left(\mathbf{y}, \left\{\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x}_i)^\top \mathbf{w}_l^*\right\}_{l=1}^L\right)$$

Although normalizations e.g. $\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x}), \mathbf{w}_l^* \in S^{D-1}$ simplify the presentation, they are neither essential to the development of the SiameseXML framework nor its analysis. This motivates the following negative log-likelihood (NLL) expression for MLE-based estimation of $\boldsymbol{\theta}^*$ and $\{\mathbf{w}_l^*\}_{l=1}^L$.

$$\mathcal{L}(\boldsymbol{\theta}, \{\mathbf{w}_l\}) \stackrel{\text{def}}{=} -\frac{1}{N}\sum_{i=1}^{N} \ln P_L\left(\mathbf{y}, \left\{\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x}_i)^\top \mathbf{w}_l^*\right\}_{l=1}^L\right) \quad (1)$$

Equation (1) subsumes objectives used by 1-vs-all methods in literature and but is prohibitive to optimize naively at extreme scales, requiring $\Omega(NLD)$ time to perform even a single gradient descent step. To remedy this, a novel postulate is introduced below that posits that for any label $l \in [L]$, if its label-text $\mathbf{z}_l$ is presented as a data point, then the label $l$ itself is likely to be found relevant. To formalize this notion, let $p_l : \{-1, +1\} \times [-1, +1] \to [0, 1]$ satisfying $p_l(+1, v) + p_l(-1, v) = 1$ for every $v \in [-1, +1]$ be some *point likelihood* functions. For example, $p_l$ could be the marginals of the joint likelihood function $P_L$.

**Postulate 2** (Label Self Proximity)**.** *For every label $l \in [L]$, its label text $\mathbf{z}_l \in \mathcal{X}$ satisfies $\mathbb{P}\left[y_l = +1 \,|\, \mathbf{z}_l, \mathbf{w}_l^*, \boldsymbol{\theta}^*\right] = p_l\left(+1, \mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{z}_l)^\top \mathbf{w}_l^*\right) \geq p_l(+1, 1) - \epsilon_l$ for some $\epsilon_l \ll 1$.*

Note that Postulate 2 always holds for some $\epsilon_l \leq 1$. However, in applications where data points and labels reside in similar spaces, Postulate 2 can be expected to hold with small values of $\epsilon_l \ll 1$. This is especially true of product-to-product recommendation tasks where data points and labels come from the same universe and a product can be reasonably expected to be strongly related to itself. However, this is also expected in other applications enumerated in Section 1 such as product-to-query and query-to-bid-phrase where the textual representations of related data point-label pairs, *i.e.,* where $y_{il} = 1$, convey similar intent.

A useful special case of the above discussion is when the joint likelihood decomposes over point likelihoods, *i.e.,*

$$P_L\left(\mathbf{y}, \left\{\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x}_i)^\top \mathbf{w}_l^*\right\}_{l=1}^L\right) = \prod_{l=1}^{L} p_l\left(y_{il}, \mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x}_i)^\top \mathbf{w}_l^*\right),$$

giving us a simplified NLL expression

$$\mathcal{L}(\boldsymbol{\theta}, \{\mathbf{w}_l\}) \stackrel{\text{def}}{=} -\frac{1}{NL}\sum_{i=1}^{N}\sum_{l=1}^{L} \ln p_l\left(y_{il}, \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i)^\top \mathbf{w}_l\right) \quad (2)$$

Decomposable likelihoods make it convenient to perform independent training of classifiers which is critical when scaling 100M labels. However, neither the SiameseXML framework nor its analysis demands decomposable likelihoods. Postulates 2 is key to the modular nature of SiameseXML since it immediately yields the following.

**Lemma 1.** *If $p_l(+1, \cdot)$ is monotonically increasing and has an inverse that is $C_{lip}$-Lipschitz over the range set $R_p \stackrel{\text{def}}{=} \{p_l(+1, v) : v \in [-1, +1]\}$, then for all $l \in [L]$ and $\mathbf{x} \in \mathcal{X}$,*

$$\left|\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x})^\top \mathbf{w}_l^* - \mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{x})^\top \mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{z}_l)\right| \leq \sqrt{2C_{lip} \cdot \epsilon_l},$$

*where $\epsilon_l$ is the label self-proximity parameter from Postulate 2. If the joint log-likelihood function $\ln P_L$ is $(q, D_{lip})$-Lipschitz for some $q \geq 1$ i.e. for every label vector $\mathbf{y} \in \{-1, 1\}^L$ and any score vectors $\mathbf{s}, \mathbf{s}' \in [-1, 1]^L$, we have $|\ln P_L(\mathbf{y}, \mathbf{s}) - \ln P_L(\mathbf{y}, \mathbf{s}')| \leq D_{lip} \cdot \|\mathbf{s} - \mathbf{s}'\|_q$, then*

$$\mathcal{L}(\boldsymbol{\theta}^*, \{\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{z}_l)\}) \leq \mathcal{L}(\boldsymbol{\theta}^*, \{\mathbf{w}_l^*\}) + D_{lip} \cdot \sqrt{2C_{lip} \cdot \epsilon_{eff}},$$

*where $\epsilon_{eff} = \left\|\left[\sqrt{\epsilon_1}, \ldots, \sqrt{\epsilon_L}\right]\right\|_q^2$. For the special case of decomposable likelihoods, this result can be clarified*

*further: if the point log-likelihood functions* $\ln p_l(+1,\cdot)$ *and* $\ln p_l(-1,\cdot)$ *are both* $D_{lip}$*-Lipschitz over the interval* $[-1,+1]$*, then we have*

$$\mathcal{L}(\boldsymbol{\theta}^*, \{\mathcal{E}_{\boldsymbol{\theta}^*}(\mathbf{z}_l)\}) \leq \mathcal{L}(\boldsymbol{\theta}^*, \{\mathbf{w}_l^*\}) + D_{lip} \cdot \sqrt{2C_{lip} \cdot \bar{\epsilon}},$$

*where* $\bar{\epsilon} \stackrel{\text{def}}{=} \frac{1}{L}\sum_{i=1}^{L} \epsilon_l$ *is the avg label self-proximity value.*

All proofs and additional details are presented in the supplementary material[1]. Lemma 1 shows that label text embeddings are capable of acting as a stand-ins for the 1-vs-all classifiers. This in turn motivates a notion of *incomplete* NLL defined over the embedding model parameters alone:

$$\mathcal{L}^1(\boldsymbol{\theta}) \stackrel{\text{def}}{=} -\frac{1}{N}\sum_{i=1}^{N} \ln P_L\left(\mathbf{y}, \left\{\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)\right\}_{l=1}^{L}\right) \quad (3)$$

Note that the expression for $\mathcal{L}^l$ naturally suggests a Siamese architecture as it requires learning a shared embedding model $\mathcal{E}_{\boldsymbol{\theta}}$ to embed both data point as well as label text, so that $\mathcal{L}^1$ is minimized. SiameseXML crucially exploits this to bootstrap the learning process. Given a solution, say $\hat{\boldsymbol{\theta}}$, that (approximately) optimizes $\mathcal{L}^1$, the 1-vs-all extreme classifiers are now reparameterized as $\mathbf{w}_l = \mathfrak{N}(\mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{z}_l) + \boldsymbol{\eta}_l)$, where $\mathfrak{N}$ is the normalization operator, *i.e.,* $\mathfrak{N} : \mathbf{v} \mapsto \mathbf{v}/\left\|\mathbf{v}\right\|_2 \in S^{D-1}$ and $\boldsymbol{\eta}_l \in \mathbb{R}^D$ are "refinement" vectors, learnt one per label and trainable using another incomplete NLL expression, this time over the refinement vectors alone.

$$\mathcal{L}^2(\{\boldsymbol{\eta}_l\}) \stackrel{\text{def}}{=} -\frac{1}{N}\sum_{n=1}^{N} \ln P_L\left(\mathbf{y}, \left\{\mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{x}_i)^\top \mathbf{w}_l\right\}_{l=1}^{L}\right) \quad (4)$$

The following result shows that this multi-stage learning process does yield embedding model parameters and 1-vs-all classifiers with bounded suboptimality with respect to the original NLL function $\mathcal{L}$ from Equation (1).

**Lemma 2.** *Suppose* $\hat{\boldsymbol{\theta}}$ *is obtained by (approximately) optimizing* $\mathcal{L}^1$ *with* $\delta_{opt}$ *being the suboptimality, i.e.,* $\mathcal{L}^1(\hat{\boldsymbol{\theta}}) \leq \min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathcal{L}^1(\boldsymbol{\theta}) + \delta_{opt}$ *and* $\hat{\mathbf{w}}_l$ *are created using refinement vectors* $\hat{\boldsymbol{\eta}}_l$ *obtained by (approximately) optimizing* $\mathcal{L}^2$ *in a monotonic manner i.e.* $\mathcal{L}^2(\{\hat{\boldsymbol{\eta}}_l\}) \leq \mathcal{L}^2(\{\mathbf{0}\})$*, then* $\mathcal{L}(\hat{\boldsymbol{\theta}}, \{\hat{\mathbf{w}}_l\}) \leq \mathcal{L}(\boldsymbol{\theta}^*, \{\mathbf{w}_l^*\}) + D_{lip} \cdot \sqrt{2C_{lip} \cdot \epsilon} + \delta_{opt}$*, where* $\epsilon$ *is the sub-optimality (either* $\epsilon_{eff}$ *or* $\bar{\epsilon}$*) in Lemma 1.*

**The 4 Modules**: The above discussion motivates a modular framework adopted by SiameseXML that specializes the more general DeepXML framework (Dahiya et al., 2021).
**Module I** (*Siamese Module*): Learn an an intermediate model $\hat{\boldsymbol{\theta}}^0$ for a Siamese architecture $\mathcal{E}_{\boldsymbol{\theta}}$ to embed data point and label texts. To do so, a objective such as the incomplete NLL $\mathcal{L}^1$ or else the triplet loss popular in Siamese training literature is minimized.
**Module II** (*Negative Mining Module*): Mine $\mathcal{O}(\log L)$ hard negative labels for each data point based on embeddings learned in Module-I.
**Module III** (*Transfer Module*): Reparameterize per-label extreme classifiers as $\mathbf{w}_l = \mathfrak{N}(\mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{z}_l) + \boldsymbol{\eta}_l)$.
**Module IV** (*Extreme Module*): Learn the refinement vectors $\boldsymbol{\eta}_l$ (and implicitly the extreme classifiers $\mathbf{w}_l$) by minimizing an objective such as the incomplete NLL $\mathcal{L}^2$ or else some classification loss like BCE. However, only terms corresponding to positive and mined negative labels from Module II are used for any data point. Optionally, the embedding model $\boldsymbol{\theta}^0$ can be jointly fine-tuned as well.

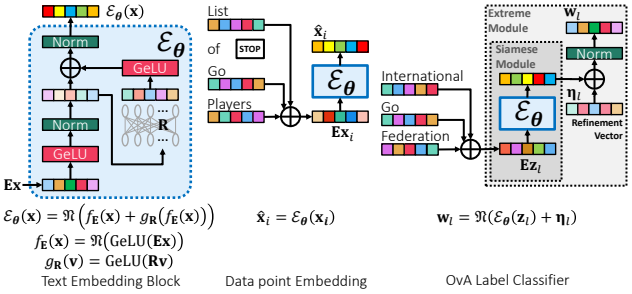


Figure 1: SiameseXML's embedding and classifier creation architectures. (best viewed under magnification).

**Recovering Siamese Models from SiameseXML**: Traditional *pure* Siamese models emerge as special cases of the above framework in a straightforward manner if we terminate the pipeline after completion of Module I itself. However, experimental results in Section 6 establish that the additional modules that enable learning of extreme classifiers on top of the Siamese model learnt in Module I greatly increase both model capacity and accuracy for XML tasks in ranking and recommendation that present few-shot labels.

## 4. Implementation Details

Various architectures and training objectives can be used to instantiate the various modules of SiameseXML. In the following, we present an implementation based on an inexpensive bag-of-embedding based encoder that has been demonstrated (Dahiya et al., 2021; Mittal et al., 2021a;b; Saini et al., 2021) to be well-suited to short-text applications and offer extremely low latency that is attractive for real-time ranking and recommendation applications.

**Architecture Details**: SiameseXML uses sparse TF-IDF vectors as input representations for both data-point and label texts. If $V$ is the number of tokens in the vocabulary, then $\mathcal{X} = \mathbb{R}^V$ and $\mathbf{x}_i, \mathbf{z}_l \in \mathbb{R}^V$. The results of Section 3 are agnostic to this choice. SiameseXML uses a frugal embedding architecture $\mathcal{E}$ parameterized as $\boldsymbol{\theta} = \{\mathbf{E}, \mathbf{R}\}$

[1]Link to Supplementary Material:
http://manikvarma.org/pubs/dahiya21b-supp.pdf

where $\mathbf{E} = \mathbb{R}^{D \times V}$ denotes $D$-dimensional embeddings $\mathbf{e}_t \in \mathbb{R}^D$ learnt for each token $t \in [V]$ in the vocabulary and $\mathbf{R} \in \mathbb{R}^{D \times D}$ represents a residual layer. Initially, an "intermediate" bag-of-embeddings embedding $f_{\mathbf{E}}(\mathbf{x}) = \mathfrak{N}(\text{GeLU}(\mathbf{E}\mathbf{x}))$ is computed ($\mathfrak{N}$ is the normalization operator, *i.e.*, $\mathfrak{N} : \mathbf{v} \mapsto \mathbf{v}/\left\|\mathbf{v}\right\|_2$) and then passed through the residual layer to get the "final" embedding $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})$ (see Figure 1). Similar architectures have been demonstrated to work well on short text applications with millisecond latencies (Dahiya et al., 2021; Mittal et al., 2021a).

**Training Objectives**: SiameseXML uses a decomposable likelihood model (see (2)) and incomplete NLL with respect to the following point-likelihood functions.

$$p_l(+1, v) = \frac{c \cdot \exp\left(d \cdot v\right)}{\exp(d)}, p_l(-1, v) = 1 - p_l(+1, v) \quad (5)$$

The scaling constants $c, d$ control the aggressiveness of the NLL function. For $c \in (0, 1), d \geq 1$, the inverse of $p_l(+1, \cdot)$ exists and is $\frac{\exp(2d)}{cd}$-Lipschitz over $R_p$, and $\ln p_l(1, \cdot), \ln p_l(-1, \cdot)$ are respectively $d$ and $\frac{cd}{1-c}$-Lipschitz over $[-1, +1]$. These calculations are detailed in the supplementary material. The corresponding training objectives $\mathcal{L}^1, \mathcal{L}^2$ contain $NL$ terms and performing optimization naively takes $\Omega\left(NDL\right)$ time for a single gradient step which is prohibitive since $N, L \geq 10^6, D \geq 10^2$. To avoid this, Modules IV optimizes $\mathcal{L}^2$ in $\mathcal{O}\left(ND \log L\right)$ time using negatives mined in Module II whereas Module I does the same for $\mathcal{L}^1$ using in-batch online negative mining.

**Module I**: In this module, parameters $\boldsymbol{\theta}$ for the Siamese model $\mathcal{E}_{\boldsymbol{\theta}}(\cdot)$ are learnt. Whereas it is common for in-batch mining strategies to create batches over data points and mine negative labels for a data point within positive labels of other data points in its mini-batch (Faghri et al., 2018; Guo et al., 2019; Chen et al., 2020; He et al., 2020), this was found to neglect rare labels and a disproportionate number of mined negatives ended up being popular labels. This not only starved rare labels of gradient updates, but also starved data points of the most useful negative labels which could be rare. SiameseXML mitigates this problem by creating batches over labels and mining negative data points instead (see Fig. 2). The Siamese nature of Module I allows such a mirrored strategy seamlessly. Label mini-batches of size $B$ were created by sampling labels from a skewed label distribution (specifically the label distribution raised to the power $0.45$) similar to (Mikolov et al., 2013). This choice was found to offer healthy representation to both popular and rare labels. For each label $l$ in a batch, a single positive data point was sampled uniformly at random and $\kappa = 1$–$5$ hard negative data points were selected from the pool of $B - 1$ positive data points for other labels in the batch. However, offline hard-negative mining can also have been used (please see the supplementary material for a discussion).

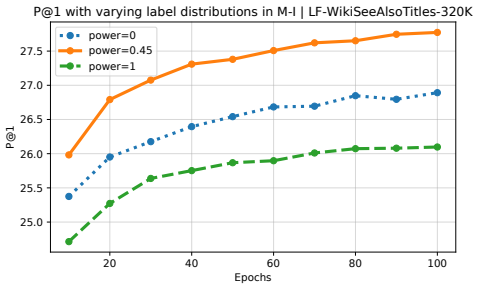


Figure 2: P@1 on LF-WikiSeeAlsoTitles-320 when label mini-batches are created in Module I using the label distribution raised to various powers. power = 0 gives the uniform distribution over labels whereas power = 1 recovers the (heavy-tailed) label distribution. Creating data-point mini-batches offers a label distribution similar to power = 1.

*Multiple Label Representations*: Although the label-text label representation itself offers superior accuracies, SiameseXML uses multiple label representations to further augment the label representation. We note that the framework in Section 3 can readily accommodate such augmented representations. For each label $l \in [L]$, the label embedding $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$ is used as well as the *label centroid* defined as $\mathbf{v}_l \stackrel{\text{def}}{=} \mathfrak{N}\left(\frac{1}{|\mathcal{R}_l|}\sum_{i \in \mathcal{R}_l} f_{\hat{\mathbf{E}}}(\mathbf{x}_i)\right)$ where $\mathcal{R}_l \stackrel{\text{def}}{=} \{i \,|\, y_{il} = 1\}$ is the set of data points for which label $l$ is relevant. Since computing $\mathbf{v}_l$ may be expensive, especially for popular labels, an estimate $\hat{\mathbf{v}}_l$ computed using two randomly sampled positive data points of that label was used instead.

*Training Objective*: For any label $l \in [L]$, let $i_l$ denote the positive data point sampled for it, $\hat{\mathbf{v}}_l$ denote its centroid estimate, $\hat{\mathcal{N}}_l^b$ denote the set of $\kappa$ in-batch hard negatives identified for that label and let $\mathcal{S}_l^b \stackrel{\text{def}}{=} \{i_l\} \cup \hat{\mathcal{N}}_l^b$. The partial NNL objective $\mathcal{L}^1$ as given by the point likelihood functions in (5) was augmented as follows to incorporate the multiple label representations and used to train Module I.

$$\hat{\mathcal{L}}^1(\boldsymbol{\theta}) \stackrel{\text{def}}{=} -\frac{\beta}{L(\kappa+1)}\sum_{l=1}^{L}\sum_{i \in \mathcal{S}_l} \ln p_l\left(y_{il}, \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) - \frac{(1-\beta)}{L(\kappa+1)}\sum_{l=1}^{L}\sum_{i \in \mathcal{S}_l^b} \ln p_l\left(y_{il}, \hat{\mathbf{v}}_l^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)) \quad (6)$$

A fixed value of $\beta = 0.7$ was used and not tuned. (6) trains both the label text embedding and the label centroid to approach the embedding of a related data point and offers improvements over using label-text embeddings alone.

**Module II**: Let $\boldsymbol{\theta}^0 = \left\{\hat{\mathbf{E}}, \hat{\mathbf{R}}^0\right\}$ be the parameters learnt by Module I by minimizing (6). To accelerate training using $\mathcal{L}^2$ in Module IV, SiameseXML mines $\mathcal{O}\left(\log L\right)$ hard negative labels for each data point. First, using a label-correlation operator $M \in \mathbb{R}^{L \times L}$ similar to one used in (Mit-

tal et al., 2021b) (see supplementary material for details), the intermediate label embeddings were linearly transformed to obtain $\tilde{\mathbf{z}}_l = \sum_{m \in [L]} M_{lm} \cdot f_{\hat{\mathbf{E}}}(\mathbf{z}_m)$. Incorporating label correlations in this inexpensive manner offered mild benefits in prediction accuracy. Three small-world (HNSW) graphs (Malkov & Yashunin, 2016) were then created: **a)** ANNS$^z$ over the transformed intermediate embeddings of the label text, *i.e.,* $\tilde{\mathbf{z}}_l$ for label $l$; **b)** ANNS$^\mu$ over label centroids $\mathbf{v}_l$ as defined earlier; and **c)** ANNS$^x$ over the intermediate embeddings of the data points $f_{\hat{\mathbf{E}}}(\mathbf{x}_i)$. Negative mining was done using $f_{\hat{\mathbf{E}}}(\cdot)$ and not $\mathcal{E}_{\boldsymbol{\theta}^0}(\cdot)$ to align training and testing since $\hat{\mathbf{E}}$ remains frozen through Module IV whereas $\mathcal{E}$ get modified in Module IV as the residual layer $\mathbf{R}$ gets fine-tuned jointly with the 1-vs-all classifiers. The three graphs were then queried a cost of $O(ND \log L)$ (time complexity derivation in the supplementary material) to generate $\mathcal{O}(\log L) \leq 500$ negative labels for each data point $i$ with three components: **a)** $\mathcal{N}_i^z = \{l \,|\, i \in \text{ANNS}^z(f_{\hat{\mathbf{E}}}(\mathbf{x}_i))\}$ **b)** $\mathcal{N}_i^\mu = \{l \,|\, \mathbf{v}_l \in \text{ANNS}^\mu(f_{\hat{\mathbf{E}}}(\mathbf{x}_i))\}$, and **c)** $\mathcal{N}_i^x = \{l \,|\, y_{jl} = 1, j \in \text{ANNS}^x(f_{\hat{\mathbf{E}}}(\mathbf{x}_i))\}$.

**Module III**: The extreme classifiers are initialized using the transformed label embeddings i.e. $\mathbf{w}_l = \mathfrak{N}(\tilde{\mathbf{z}}_l + \boldsymbol{\eta}_l)$.

**Module IV**: SiameseXML jointly trains the refinement vectors $\boldsymbol{\eta}_l$ and fine-tunes the residual layer $\mathbf{R}$ (token embeddings $\hat{\mathbf{E}}$ learnt in Module I remain frozen) restricted to the set $\mathcal{S}_i \overset{\text{def}}{=} \mathcal{P}_i \cup \hat{\mathcal{N}}_i$ for each data point $i \in [N]$ where $\mathcal{P}_i = \{l \,|\, y_{il} = 1\}$ is the set of positive labels for data point $i$ and $\hat{\mathcal{N}}_i$ is the set of $\kappa \cdot |\mathcal{P}_i|$ hardest negatives from the label shortlist $\hat{\mathcal{N}}_i^z \cup \hat{\mathcal{N}}_i^\mu \cup \hat{\mathcal{N}}_i^x$ obtained in Module-II ($\kappa = 2$ – 5). Recall that data points in XML applications typically have $\mathcal{O}(\log L)$ positive labels i.e. $|\mathcal{P}_i| \leq \mathcal{O}(\log L)$. Since $\left|\hat{\mathcal{N}}_i\right| \leq \mathcal{O}(\log L)$ by design, we have $|\mathcal{S}_i| \leq \mathcal{O}(\log L)$. The following objective, that contains only $\mathcal{O}(N \log L)$ terms, was minimized using mini-batches over data points and the Adam optimizer (Kingma & Ba, 2015)

$$\hat{\mathcal{L}}^2(\{\boldsymbol{\eta}_l\}, \mathbf{R}) \overset{\text{def}}{=} -\frac{1}{N \log L} \sum_{n=1}^{N} \sum_{l \in \mathcal{S}_i} \ln p_l \left(y_{il}, \mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{x}_i)^\top \mathbf{w}_l\right)$$

to obtain the 1-vs-all classifiers $\hat{\mathbf{H}} = [\hat{\boldsymbol{\eta}}_1, \ldots, \hat{\boldsymbol{\eta}}_L]$ and a fine-tuned embedding model $\hat{\boldsymbol{\theta}} = \{\hat{\mathbf{E}}, \hat{\mathbf{R}}\}$.

**Alternate Training Objectives**: Experiments in the supplementary material show SiameseXML using alternate loss functions (e.g. triplet loss for Module I and BCE loss for Module IV) also offers stable performance. However, the triplet loss is unsuitable for Module IV whereas BCE is ill-suited for Module I. In contrast, the (incomplete) NLLs created using the point likelihood functions $p_l$ offer the convenience of using a single formulation across the modules.

**Asynchronous distributed training on 100M labels**: Optimizing $\hat{\mathcal{L}}^2$ on a single GPU was infeasible for the largest dataset Q2B-100M given the $\Omega(LD)$ memory footprint needed to train all $L$ 1-vs-all classifiers. Multiple GPUs were deployed to speed up training by partitioning the 1-vs-all classifiers $\{\mathbf{w}_l\}$ into subsets $\{\mathbf{w}_l\}^1, \ldots \{\mathbf{w}_l\}^g$, each subset being trained on a separate GPU. As $\mathbf{R}$ is jointly trained in this module and shared across labels, this brought up synchronization issues. SiameseXML eliminated the need for a high-bandwidth connection between GPUs and effected entirely asynchronous training on completely disconnected machines by learning a separate residual $\mathbf{R}^j$ for each label subset using the slightly modified objective

$$\hat{\mathcal{L}}_j^2(\{\boldsymbol{\eta}_l\}, \mathbf{R}^j) \overset{\text{def}}{=} -\frac{1}{NL} \sum_{n=1}^{N} \sum_{l \in \mathcal{S}_i^j} \ln p_l \left(y_{il}, \mathcal{E}_{\hat{\boldsymbol{\theta}}}^j(\mathbf{x}_i)^\top \mathbf{w}_l\right).$$

where $j$ is the id of the GPU, $\mathcal{S}_i^j = \mathcal{S}_i \cap \{\mathbf{w}_l\}^j$, and $\mathcal{E}_{\hat{\boldsymbol{\theta}}}^j(\mathbf{x}_i)$ is the embedding function using $\mathbf{R}^j$ as the residual. Using $\hat{\mathcal{L}}_j^2$ instead of $\hat{\mathcal{L}}^2$ was found to yield comparable accuracy.

**Log-time Prediction**: Given a test point $\mathbf{x} \in \mathbb{R}^V$, iterating over all $L$ 1-vs-all classifiers takes $\mathcal{O}(DL)$ time which is prohibitive when $L \approx 10^8, D \geq 10^2$. SiameseXML makes use of the ANNS graphs from Module-II to accelerate inference in 3 steps: **a)** extract the intermediate $f_{\hat{\mathbf{E}}}(\mathbf{x})$ and final $\mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{x})$ embeddings for the test point; **b)** shortlist $\mathcal{O}(\log L)$ of the most relevant labels as $\mathcal{S} = \{l \,|\, l \in \text{ANNS}^z(f_{\hat{\mathbf{E}}}(\mathbf{x})) \cup \text{ANNS}^\mu(f_{\hat{\mathbf{E}}}(\mathbf{x}))\} \cup \{l \,|\, y_{jl} = 1, j \in \text{ANNS}^x(f_{\hat{\mathbf{E}}}(\mathbf{x}))\}$; and finally **c)** evaluate 1-vs-all classifiers for only the shortlisted labels to get the scores $\hat{y}_l = \alpha \cdot \mathbf{w}_l^\top \mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) + (1 - \alpha) \cdot (f_{\hat{\mathbf{E}}}(\mathbf{z}_l) + \mathbf{v}_l)^\top \mathcal{E}_{\hat{\boldsymbol{\theta}}}(\mathbf{x})$ if label $l \in \mathcal{S}$ and $\hat{y}_l = -\infty$ otherwise, where $\alpha \in [0, 1]$ is a hyper-parameter set via validation. This allowed predictions in $\mathcal{O}\left(D^2 + D \log L\right)$ time (derivation in the supplementary material) and in practice, translated to around 12 milliseconds on a CPU even on the dataset with 100M labels.

## 5. Generalization Bounds

Generalization bounds for multi-label problems incur either an explicit dependence on the number of labels $L$ (Zhang, 2004) or else an implicit one via regularization constants or norms of 1-vs-all classifiers (Bartlett et al., 2017; Yu et al., 2014). However, Module I of SiameseXML learns parameters $\boldsymbol{\theta}$ that are of size independent of $L$, and yet yields a fully functional classifier model, namely $\hat{\mathbf{w}}_l^0 = \mathcal{E}_{\hat{\boldsymbol{\theta}}^0}(\mathbf{z}_l)$ owing to Lemma 1. This presents an opportunity to establish bounds that avoid any dependence on $L$, explicit or implicit. Theorem 3 (Part I) establishes such a result.

On the other hand, contemporary generalization bounds for deep networks (Bartlett et al., 2017; Golowich et al., 2018; Long & Sedghi, 2020; Wei & Ma, 2019) do not take label metadata into account and thus, their results do not directly apply to Siamese architectures. Moreover, such an application is expected to yield bounds for Module I that depend on

the Frobenius norm of $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_L] \in \mathbb{R}^{V \times L}$ that can scale as $\Omega(\sqrt{L})$. This is because the proof techniques such as those used in (Bartlett et al., 2017) rely on "empirical" covering numbers which need only cover a bounded set and can hence use weaker inequalities such as the Chevyshev's inequality within the standard "empirical" Maurey lemma. To avoid a dependence on $L$, Theorem 3 instead relies on uniform covers of (infinite) compact subsets. The Chevyshev's inequality no longer suffices and a novel *uniform* Maurey-type sparsification lemma is established using Bernstein bounds over Hilbert spaces.

Theorem 3 depends on the token embeddings $\mathbf{E}$ as $\sqrt{\|\mathbf{E}\|_{\infty,1} \|\mathbf{E}\|_{1,1}} \leq \|\mathbf{E}\|_{1,1}$ whereas (Bartlett et al., 2017) incur a dependence on the mixed norm $\left\|\mathbf{E}^\top\right\|_{2,1}$ (see below for notation as well as note that (Bartlett et al., 2017) would have styled the matrix $\mathbf{E}$ as $\mathbf{E}^\top$). This indicates a suboptimality of upto $\sqrt{D}$. However, in XML settings, small $D \approx 100$, *i.e.,* $\sqrt{D} \approx 10$ are common which suggests that Theorem 3 presents a favorable trade-off as it avoids a dependence on $L$, *e.g.* via $\|\mathbf{Z}\|_F$ which could scale as $\Omega\left(\sqrt{L}\right)$, in favor of an extra $\sqrt{D}$ factor. Theorem 3 (Part II) extends to the model learnt in Module IV that incorporates extreme classifiers. This bound is no longer independent of $L$ but depends on it explicitly in a weak manner via $\log L$ and implicitly via the $L_{1,1}$ norm of the refinement vector matrix $\mathbf{H} = [\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_L] \in \mathbb{R}^{D \times L}$. It would be interesting to tighten these bounds, for instance using techniques from (Long & Sedghi, 2020; Golowich et al., 2018).

**Notation**: For a label vector $\mathbf{y} \in \{-1, +1\}^L$, let $P_\mathbf{y} := \{l : y_l = +1\}$ and $N_\mathbf{y} := \{l : y_l = -1\}$ denote the sets of positive and negative labels respectively. Given a score vector $\mathbf{s} = [s_1, \ldots, s_L] \in [-1, 1]^L$, let $\pi_\mathbf{s} \in \text{Sym}([L])$ be the permutation that ranks labels in decreasing order of their scores according to $\mathbf{s}$ i.e. $s_{\pi_\mathbf{s}(1)} \geq s_{\pi_\mathbf{s}(2)} \geq \ldots$. We will also let $\pi^+_\mathbf{s} \in \text{Sym}(P_\mathbf{y})$ denote the permutation that ranks the positive labels in decreasing order of their scores according to $\mathbf{s}$ i.e. $\pi^+_\mathbf{s}(t) \in P_\mathbf{y}$ for all $t \in |P_\mathbf{y}|$. For any $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\|\mathbf{A}\|_\sigma := \sup_{\mathbf{x} \in \mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$ denotes its spectral norm. For $p, q \in [1, \infty]$, define the mixed norm $\|\mathbf{A}\|_{p,q} := \left\| \left[ \|\mathbf{A}_{1,:}\|_p, \|\mathbf{A}_{2,:}\|_p, \ldots, \|\mathbf{A}_{m,:}\|_p \right] \right\|_q$. This first takes the $p$-th norm over all rows, then the $q$-th norm. To be sure, this is slightly different from popular convention that takes norms over columns first. However, this change of convention avoids clutter due to repeated transpose symbols.

**Definitions**: The prec@k loss counts the fraction of top ranks not occupied by relevant labels $\wp_k(\mathbf{s}, \mathbf{y}) \stackrel{\text{def}}{=} 1 - \frac{1}{k} \sum_{t=1}^k \mathbb{I}\{\pi_\mathbf{s}(t) \in P_\mathbf{y}\}$. Given a margin $\gamma > 0$, define the $\gamma$-ramp function as $r_\gamma(v) \stackrel{\text{def}}{=} \min\left\{\max\left\{\frac{v}{\gamma}, 0\right\}, 1\right\}$. For $k \in \mathbb{N}$, let the surrogate prec@k loss be $\ell^{\text{prec}}_{\gamma,k}(\mathbf{s}, \mathbf{y}) \stackrel{\text{def}}{=} 1 - \frac{1}{k} \sum_{t=1}^{\min\{k, |P_\mathbf{y}|\}} r_\gamma\left(s_{\pi^+_\mathbf{s}(t)} - \max_{l' \in N_\mathbf{y}} s_{l'}\right)$. For any data point $\mathbf{x} \in \mathbb{R}^V$ and model $\boldsymbol{\theta}$, denote $s^{\boldsymbol{\theta}}(\mathbf{x}) = [s^{\boldsymbol{\theta}}_1, \ldots, s^{\boldsymbol{\theta}}_L]^\top \in \mathbb{R}^L$, where $s^{\boldsymbol{\theta}}_l \stackrel{\text{def}}{=} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$ are scores assigned to various labels for this data point using the Siamese model obtained after Module I. For models obtained after Module IV that incorporate extreme classifiers $\mathbf{H} = [\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_L]$, scores are redefined as $s^{\boldsymbol{\theta},\mathbf{H}}_l \stackrel{\text{def}}{=} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})^\top \mathbf{w}_l$. For training data sampled as $(\mathbf{x}_i, \mathbf{y}_i) \sim \mathcal{D}, i \in [N]$, the surrogate empirical risk for a model $\boldsymbol{\theta}$ is $\hat{\ell}_N(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^N \ell^{\text{prec}}_{\gamma,k}(s^{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i)$. The population prec@k risk for a model $\boldsymbol{\theta}$ is $\wp_k(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}} \left[\wp_k(s^{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y})\right]$. $\ell(\boldsymbol{\theta}, \mathbf{H})$ and $\hat{\ell}_N(\boldsymbol{\theta}, \mathbf{H})$ are similarly defined using $s^{\boldsymbol{\theta},\mathbf{H}}_l$ instead.

**Theorem 3.** *(Part I) If $\hat{\boldsymbol{\theta}}^0$ is the model obtained after Module I and $\|\mathbf{x}_i\|_0, \|\mathbf{z}_l\|_0 \leq s$, then with prob. $1 - \delta$,*

$$\wp_k(\hat{\boldsymbol{\theta}}^0) \leq \hat{\ell}_N(\hat{\boldsymbol{\theta}}^0) + \frac{1}{\gamma} \cdot \frac{P \ln(N)}{\sqrt{N}} + \sqrt{\frac{\ln \frac{1}{\delta}}{N}},$$

*where $P = \mathcal{O}\left(\sqrt{D \ln(D)} \sqrt{R^R_\infty R^R_1} + \sqrt{s \ln(DV)} R^R_\sigma \sqrt{R^E_\infty R^E_1}\right)$ and $R^E_1 = \|\hat{\mathbf{E}}\|_{1,1}, R^E_\infty = \|\hat{\mathbf{E}}\|_{\infty,1}, R^R_1 = \|\hat{\mathbf{R}}^0\|_{1,1}, R^R_\infty = \|\hat{\mathbf{R}}^0\|_{\infty,1}, R^R_\sigma = \|\hat{\mathbf{R}}^0\|_\sigma$*

*(Part II) If $\left\{\hat{\boldsymbol{\theta}}, \hat{\mathbf{H}}\right\}$ is the model after Module IV ($\hat{\mathbf{E}}$ from Module I, $\hat{\mathbf{R}}, \hat{\mathbf{H}}$ learnt in module IV). Then w.p. $1 - \delta$,*

$$\wp_k(\hat{\boldsymbol{\theta}}, \hat{\mathbf{H}}) \leq \hat{\ell}_N(\hat{\boldsymbol{\theta}}, \hat{\mathbf{H}}) + \frac{1}{\gamma} \cdot \frac{Q \ln(N)}{\sqrt{N}} + \sqrt{\frac{\ln \frac{1}{\delta}}{N}},$$

*where $Q = P + \mathcal{O}\left(\ln(DL) \sqrt{R^C_\infty R^C_1}\right), R^C_1 = \|\hat{\mathbf{H}}\|_{1,1}, R^C_\infty = \|\hat{\mathbf{H}}\|_{1,\infty}$ and $R^R_1 = \|\hat{\mathbf{R}}\|_{1,1}, R^R_\infty = \|\hat{\mathbf{R}}\|_{\infty,1}, R^R_\sigma = \|\hat{\mathbf{R}}\|_\sigma$.*

The supplementary material contains a more relaxed discussion of related work and a proof for Theorem 3.

## 6. Experiments

**Datasets**: Multiple benchmark short-text and long-text datasets for product-to-product recommendation as well as predicting related Wikipedia articles were considered. The datasets and provenance details thereof can be found on the Extreme Classification Repository (Bhatia et al., 2016). Results are also reported on proprietary datasets with up to 100 million labels for matching user queries to advertiser bid phrases (Q2BP-4M, Q2BP-40M, and Q2BP-100M). These were created by mining click logs of the Bing search engine where a query was treated as a data point and clicked advertiser bid phrases became its labels. The Q2BP-40M and Q2BP-100M datasets were created by mining logs of different streams, whereas Q2BP-4M was created by randomly sampling 10% labels from Q2BP-40M to allow experimentation with less scalable methods. Please refer to the supplementary material for a discussion on the datasets.

Table 1: **Short-text datasets**: SiameseXML and the SiameseXML++ variant are both significantly more accurate than leading deep XML methods. Results are presented only for methods that converged within the timeout.

| Method | PSP@1 | PSP@3 | PSP@5 | P@1 | P@5 | Training Time (hr) |
|---|---|---|---|---|---|---|
| LF-AmazonTitles-131K | | | | | | |
| SiameseXML | **35.44** | **40.7** | **45.98** | **40.98** | **20.09** | 0.94 |
| SiameseXML++ | **35.8** | **40.96** | **46.19** | **41.42** | **21.21** | 1.13 |
| ECLARE | 33.51 | 39.55 | 44.7 | 40.74 | 19.88 | 2.16 |
| GalaXC | 32.5 | 38.79 | 43.95 | 39.17 | 19.49 | 0.42 |
| DECAF | 30.85 | 36.44 | 41.42 | 38.40 | 18.65 | 2.16 |
| X-Transformer | 21.72 | 24.42 | 27.09 | 29.95 | 13.07 | 64.40 |
| Astec | 29.22 | 34.64 | 39.49 | 37.12 | 18.24 | 1.83 |
| AttentionXML | 23.97 | 28.60 | 32.57 | 32.25 | 15.61 | 20.73 |
| LightXML | 25.67 | 31.66 | 36.44 | 35.6 | 17.45 | 71.4 |
| MACH | 24.97 | 30.23 | 34.72 | 33.49 | 16.45 | 3.30 |
| LF-WikiSeeAlsoTitles-320K | | | | | | |
| SiameseXML | **27.05** | **28.42** | **30.37** | **31.3** | **15.92** | 1.59 |
| SiameseXML++ | **26.82** | **28.42** | **30.36** | **31.97** | **16.24** | 1.87 |
| ECLARE | 22.01 | 24.23 | 26.27 | 29.35 | 15.05 | 13.46 |
| GalaXC | 19.77 | 22.25 | 24.47 | 27.87 | 14.3 | 1.08 |
| DECAF | 16.73 | 18.99 | 21.01 | 25.14 | 12.86 | 11.16 |
| Astec | 13.69 | 15.81 | 17.50 | 22.72 | 11.43 | 4.17 |
| AttentionXML | 9.45 | 10.63 | 11.73 | 17.56 | 8.52 | 56.12 |
| LightXML | 11.34 | 13.51 | 15.3 | 21.93 | 11.01 | 159 |
| MACH | 9.68 | 11.28 | 12.53 | 18.06 | 8.99 | 8.23 |
| LF-AmazonTitles-1.3M | | | | | | |
| SiameseXML | **27.93** | **31.03** | **33.00** | 47.22 | 36.96 | 8.6 |
| SiameseXML++ | **27.12** | **30.43** | **32.52** | 49.02 | 38.52 | 9.89 |
| ECLARE | 23.43 | 27.90 | 30.56 | 50.14 | 40.00 | 70.59 |
| GalaXC | 25.22 | 29.12 | 31.44 | 49.81 | 40.12 | 9.55 |
| DECAF | 22.07 | 26.54 | 29.30 | **50.67** | **40.35** | 74.47 |
| Astec | 21.47 | 25.41 | 27.86 | 48.82 | 38.44 | 18.54 |
| AttentionXML | 15.97 | 19.90 | 22.54 | 45.04 | 36.25 | 380.02 |
| MACH | 9.32 | 11.65 | 13.26 | 35.68 | 28.35 | 60.39 |

**Baselines**: Comparisons are presented against leading deep XML methods such as X-Transformer (Chang et al., 2020a), ECLARE (Mittal et al., 2021b), GalaXC (Saini et al., 2021), Astec (Dahiya et al., 2021), MACH (Medini et al., 2019), DECAF (Mittal et al., 2021a), and AttentionXML (You et al., 2019), as well as classical methods such as DiSMEC (Babbar & Schölkopf, 2017), Slice (Jain et al., 2019), and Parabel (Prabhu et al., 2018b), *etc*. X-Transformer, ECLARE, GalaXC and DECAF are of particular interest as they make use of label text to improve their predictions. Implementations provided by respective authors were used in all cases. All methods were offered a timeout of one week on a single GPU. Results are also reported for popular Siamese Networks for ad retrieval such as TwinBert (Lu et al., 2020) and CDSSM (Huang et al., 2013) on the Q2BP datasets. Please refer to the supplementary material for SiameseXML's hyper-parameter settings.

**Evaluation metrics**: Performance was evaluated using standard XML performance measures: precision@$k$ (P@$k$, $k \in \{1, 5\}$) and propensity scored precision@$k$ (PSP@$k$, $k \in \{1, 3, 5\}$). Results on nDCG@$k$ (N@$k$) and propensity scored nDCG@$k$ (PSN@$k$) are presented in the supplementary material which also contains the definitions of all these metrics. All training times are reported on a 24-core Intel Xeon 2.6 GHz machine with a single Nvidia V100 GPU. However, Q2BP datasets were afforded multiple GPUs.

Table 2: **Full-text datasets**: SiameseXML and SiameseXML++ continue to outperform leading deep XML methods including those that utilize label-text and label-graphs, such as ECLARE and GalaXC. Results are presented only for methods that converged within the timeout.

| Method | PSP@1 | PSP@3 | PSP@5 | P@1 | P@5 | Training Time (hr) |
|---|---|---|---|---|---|---|
| LF-Amazon-131K | | | | | | |
| SiameseXML | **37.09** | **43.27** | **49.37** | **44.15** | **21.73** | 0.98 |
| SiameseXML++ | **37.56** | **43.69** | **49.75** | **44.81** | **21.94** | 1.18 |
| ECLARE | 34.98 | 42.38 | 48.53 | 43.56 | 21.57 | 2.15 |
| GalaXC | 35.10 | 41.18 | 46.38 | 41.46 | 20.25 | 0.45 |
| DECAF | 34.52 | 41.14 | 47.33 | 42.94 | 21.00 | 1.80 |
| AttentionXML | 32.92 | 39.51 | 45.24 | 42.90 | 20.97 | 50.17 |
| Astec | 32.95 | 39.42 | 45.30 | 42.22 | 20.85 | 3.05 |
| LightXML | 30.27 | 37.71 | 44.10 | 41.49 | 20.75 | 56.03 |
| MACH | 25.27 | 30.71 | 35.42 | 34.52 | 17.00 | 13.91 |
| LF-WikiSeeAlso-320K | | | | | | |
| SiameseXML | **28.91** | **32.67** | **35.97** | **40.67** | **20.96** | 1.94 |
| SiameseXML++ | **29.01** | **32.68** | **36.03** | **42.16** | **21.35** | 2.33 |
| ECLARE | 26.04 | 30.09 | 33.01 | 40.58 | 20.14 | 9.40 |
| GalaXC | 25.78 | 29.37 | 32.53 | 38.96 | 19.58 | 1.10 |
| DECAF | 25.72 | 30.93 | 34.89 | 41.36 | 21.38 | 13.40 |
| Astec | 23.41 | 28.08 | 31.92 | 40.07 | 20.36 | 6.39 |
| AttentionXML | 22.67 | 26.66 | 29.83 | 40.50 | 19.87 | 90.37 |
| LightXML | 17.85 | 21.26 | 24.16 | 34.50 | 16.83 | 249.00 |
| MACH | 13.11 | 15.28 | 16.93 | 27.18 | 12.89 | 50.22 |

**Offline results**: Tables 1 and 2 present results on benchmark datasets for short- and full-text XML tasks where SiameseXML could be 2–13% more accurate than methods which make use of label meta-data, *viz.* ECLARE, GalaXC, DECAF and X-Transformer in propensity scored precision. This indicates that SiameseXML's gains are more prominent on rare labels which are more rewarding in real-world scenarios. SiameseXML could also be up to 2–68$\times$ faster at training than DECAF, ECLARE and X-Transformer. Moreover, SiameseXML was found to be 2–13% more accurate than Astec, AttentionXML, and MACH indicating that SiameseXML can be more accurate and simultaneously faster to train as compared to leading deep extreme classifiers. Moreover, SiameseXML could also be up to 2% more accurate in vanilla precision as compared to the second best method. Table 5 includes a qualitative analysis of SiameseXML's predictions. SiameseXML could also be up to 17% more accurate than leading XML methods with fixed features including Parabel, DiSMEC, Bonsai, and Slice (Please refer to the supplementary material). Results are also reported for SiameseXML++ that jointly trains token embeddings $\mathbf{E}$ in Module IV as well and offers moderate improvements over SiameseXML. Table 3 demonstrates that SiameseXML could be 10–30% more accurate than leading XML techniques that could scale to the Q2BP datasets.

Table 3: **User query to advertiser bid-phrase prediction**: SiameseXML could be significantly more accurate than leading extreme classifiers as well as Siamese architectures including TwinBert and CDSSM.

| Method | PSP@1 | PSP@3 | PSP@5 | P@1 | P@3 | P@5 |
|---|---|---|---|---|---|---|
| Q2BP-100M | | | | | | |
| SiameseXML | **74.82** | **84.28** | **87.36** | **80.84** | **40.33** | **26.71** |
| TwinBert | 60.76 | 66.58 | 69.37 | 61.54 | 30.51 | 20.42 |
| Q2BP-40M | | | | | | |
| SiameseXML | **58.86** | **66.44** | **70.59** | **70.33** | **48.13** | **37.38** |
| TwinBert | 52.44 | 55.13 | 56.43 | 57.69 | 41.53 | 33.2 |
| CDSSM | 43.41 | 42.57 | 42.83 | 48.6 | 32.29 | 25.19 |
| Parabel | 36.04 | 41.52 | 44.84 | 47.71 | 36.64 | 30.45 |
| Q2BP-4M | | | | | | |
| SiameseXML | **67.24** | **72.82** | **76.03** | **69.53** | **38.58** | **27.12** |
| CDSSM | 44.95 | 48.19 | 50.78 | 45.58 | 24.62 | 17.38 |
| Astec | 57.78 | 69.16 | 73.84 | 66.39 | 37.35 | 26.52 |
| SLICE+CDSSM | 47.41 | 59.52 | 65.89 | 54.77 | 33.15 | 24.11 |
| Parabel | 55.18 | 66.08 | 71.21 | 61.61 | 36.36 | 26.01 |

Most existing techniques could not scale to 100 million labels whereas SiameseXML could be trained within 2 days on 6× Nvidia V100 GPUs. SiameseXML was also found to be 6–21% more accurate than Siamese networks including TwinBert and CDSSM on the Q2BP datasets. This demonstrates that SiameseXML can scale to datasets with 100M labels while still being more accurate than both Siamese networks and extreme classifiers.

**Online live deployment**: SiameseXML was deployed on the Bing search engine to perform A/B tests on live search engine traffic for matching user entered queries to advertiser bid phrases (Q2BP) and was compared to an ensemble of leading (proprietary) information retrieval, XML, generative and graph based techniques. Performance was measured in terms of Click-Yield (CY), Query Coverage (QC), and Click-Through Rate (CTR). Click Yield (CY) is defined as the number of clicks per unit search query (please refer to the supplementary material for details). SiameseXML was found to increase CY and CTR by 1.4% and 0.6%, respectively. This indicates that ads surfaced using the proposed method are more relevant to the end user. Additionally, SiameseXML offered higher QC by 2.83%, indicating its ability in surfacing ads for queries where ads were previously not shown. Further, human labelling by expert judges was also performed. A random set of (query, predicted bid-phrase) pairs from various algorithms was presented to judges who labeled each pair as good or bad. SiameseXML could increase the fraction of good predictions by 11% over state-of-the-art in-production techniques.

**Ablations**: The aim of ablation experiments was to investigate SiameseXML's design choices in Modules I, II, and III. First, the sampling strategy must find a right balance between popular labels and data-scarce tail labels (See Fig. 2 for results with various sampling distributions). Second, label shortlists in Module-II ($\hat{\mathcal{N}}_i$) could be up to 4% and 10% more accurate than the shortlist computed solely based on label embeddings ($\hat{\mathcal{N}}_i^z$) in terms of precision and recall respectively (see Table 4) with comparatively larger gains on large datasets. This demonstrates that the label text by itself may not be as informative for some labels and using multiple representations for each label can lead to performance gains, especially in short-text applications. As noted earlier, Lemma 1 shows that Module I itself yields a fully functional classifier model with bounded sub-optimality. Thus, in principle, SiameseXML could stop training after Module-I and use the label text embeddings as classifiers along with the label shortlist $\hat{\mathcal{N}}_i^z$ to make final predictions. However, this is sub-optimal and training an extreme classifier in Module-III can lead to 2-6% more accurate predictions with comparatively larger gains on larger datasets indicating the utility of extreme classifiers (see Table 4).

Table 4: Impact of incrementally adding components, *i.e.,* label embedding ($f_{\hat{\mathbf{E}}}(\mathbf{z}_l)$), label centroids ($\mathbf{v}_l$) and label classifiers $\mathbf{w}_l$ on the LF-WikiSeeAlsoTitles-320K dataset.

| Components | P@1 | Recall@500 |
|---|---|---|
| $f_{\hat{\mathbf{E}}}(\mathbf{z}_l)$ | 27.6 | 57.66 |
| $\{f_{\hat{\mathbf{E}}}(\mathbf{z}_l), \mathbf{v}_l\}$ | 28.03 | 63.74 |
| SiameseXML | 31.3 | 63.74 |

Table 5: SiameseXML's predictions for the data point "**List of Go players**" from LF-WikiSeeAlsoTitles-320K are more accurate than leading methods. Mispredictions in light gray.

| Method | Predictions |
|---|---|
| SiameseXML | Go professional, List of Go organizations, List of top title holders in Go, Go ranks and ratings, List of professional Go tournaments |
| DECAF | Go players, List of Go organizations, Music of the Republic of Macedonia, Players, List of all-female bands |
| AttentionXML | List of NHL players, List of professional Go tournaments, List of foreign NBA players, List of chess grandmasters, List of Israeli chess players |

## Acknowledgements

The authors thank the anonymous reviewers for several helpful comments. The authors also thank Anshul Mittal and Praneeth Netrapalli for helpful discussions and feedback. Thanks are due to Kushal Dave, Prasenjit Dey for their help in creating the Q2BP datasets and running the A/B tests. KD thanks support from Microsoft Research India.

## References


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

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

Babbar, R. and Schölkopf, B. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108, 2019.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. Spectrally-normalized margin bounds for neural networks. In *NIPS*, 2017.

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

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

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

Chang, W.-C., Yu, F.-X., Chang, Y.-W., Yang, Y., and Kumar, S. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *ICLR*, 2020b.

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

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

Faghri, F., Fleet, D.-J., Kiros, J.-R., and Fidler, S. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*, 2018.

Golowich, N., Rakhlin, A., and Shamir, O. Size-Independent Sample Complexity of Neural Networks. In *COLT*, 2018.

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

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

He, K., Fan, H., W., Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

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

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

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

Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., and Hullermeier, E. Extreme F-measure Maximization using Sparse Probability Estimates. In *ICML*, 2016.

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

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

Khandagale, S., Xiao, H., and Babbar, R. Bonsai - Diverse and Shallow Trees for Extreme Multi-label Classification. *CoRR*, 2019.

Kingma, P. D. and Ba, J. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

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

Long, P. M. and Sedghi, H. Generalization bounds for deep convolutional neural networks. In *ICLR*, 2020.

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

Luan, Y., Eisenstein, J., Toutanova, K., and Collins, M. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics*, 9:723–729, 2020.

Malkov, A. Y. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *CoRR*, 2016.

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

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

Mineiro, P. and Karampatziakis, N. Fast Label Embeddings via Randomized Linear Algebra. In *ECML/PKDD*, 2015.

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

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

Niculescu-Mizil, A. and Abbasnejad, E. Label Filters for Large Scale Multilabel Classification. In *AISTATS*, 2017.

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

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

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

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

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

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

Tagami, Y. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *KDD*, 2017.

Wei, C. and Ma, T. Data-dependent Sample Complexity of Deep Neural Networks via Lipschitz Augmentation. In *NeurIPS*, 2019.

Wu, C.-Y., Manmatha, R., Smola, A.-J., and Krahenbuhl, P. Sampling Matters in Deep Embedding Learning. In *ICCV*, 2017.

Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., and Dembczynski, K. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NIPS*, 2018.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

Xu, C., Tao, D., and Xu, C. Robust Extreme Multi-label Learning. In *KDD*, 2016.

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

Yeh, C. K., Wu, C. W., Ko, J. W., and Wang, F. C. Y. Learning Deep Latent Spaces for Multi-Label Classification. *CoRR*, 2017.

Yen, E. I., Huang, X., Zhong, K., Ravikumar, P., and Dhillon, I. S. PD-Sparse: A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. In *ICML*, 2016.

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

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

Yu, H., Jain, P., Kar, P., and Dhillon, I. S. Large-scale Multi-label Learning with Missing Labels. In *ICML*, 2014.

Zhang, T. Statistical Analysis of Some Multi-Category Large Margin Classification Methods. *Journal of Machine Learning Research*, 5:1225–1251, October 2004.