Deep Encoders with Auxiliary Parameters for Extreme Classification

Kunal Dahiya[*]
kunalsdahiya@gmail.com
IIT Delhi
India

Sachin Yadav
t-sacyadav@microsoft.com
Microsoft Research
India

Sushant Sondhi
sushantsondhi@gmail.com
IIT Delhi
India

Deepak Saini
desaini@microsoft.com
Microsoft
USA

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

Jian Jiao
Jian.Jiao@microsoft.com
Microsoft
USA

Sumeet Agarwal
sumeet@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

The task of annotating a data point with labels most relevant to it from a large universe of labels is referred to as Extreme Classification (XC). State-of-the-art XC methods have applications in ranking, recommendation, and tagging and mostly employ a combination architecture comprised of a deep encoder and a high-capacity classifier. These two components are often trained in a modular fashion to conserve compute. This paper shows that in XC settings where data paucity and semantic gap issues abound, this can lead to sub-optimal encoder training which negatively affects the performance of the overall architecture. The paper then proposes a lightweight alternative DEXA that augments encoder training with auxiliary parameters. Incorporating DEXA into existing XC architectures requires minimal modifications and the method can scale to datasets with 40 million labels and offer predictions that are up to 6% and 15% more accurate than embeddings offered by existing deep XC methods on benchmark and proprietary datasets, respectively. The paper also analyzes DEXA theoretically and shows that it offers provably superior encoder training than existing Siamese training strategies in certain realizable settings. Code for DEXA is available at https://github.com/Extreme-classification/dexa.

CCS CONCEPTS

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

[*]Part of work done while the author was at Microsoft

KEYWORDS

Extreme multi-label learning; large-scale learning; deep encoders; sponsored search; product recommendation

ACM Reference Format:
Kunal Dahiya, Sachin Yadav, Sushant Sondhi, Deepak Saini, Sonu Mehta, Jian Jiao, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2023. Deep Encoders with Auxiliary Parameters for Extreme Classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599301

1 INTRODUCTION

Overview: Given a data point and a large universe of labels, the task of identifying the subset (*i.e.,* one or more) of labels most relevant to that data point is referred to as Extreme Classification (XC). It is notable that XC is distinct from, and indeed generalizes, multi-class classification where the data point must be annotated with a single label or class. Several tasks in ranking, recommendation and tagging applications can be cast as XC problems and several applications of this primitive have been identified in recent years in areas such as recommending related products [15, 33, 36], document categorization and tagging [2, 9, 60], search and online advertisement [14, 15, 22, 41], and recommendation related queries in search engines [9, 21]. A notable class of applications are *short-text applications* where data points and labels are endowed with short textual descriptions containing 3-10 tokens. Applications in areas such as web-search, query rewriting, related-query recommendation, related-product recommendation and product search fall in this category. In these applications, data points and labels are product titles, webpage titles or user queries which are indeed short pieces of text. These applications have attracted increased interest in recent years [9, 13, 15, 33, 36, 48].

General Challenges in XC: There are certain challenges presented by most XC settings. Most applications where XC is applied require real-time responses and thus, the set of relevant labels for a test data point must be identified within milliseconds. Similarly, in most applications where the universe of labels is in the millions, an

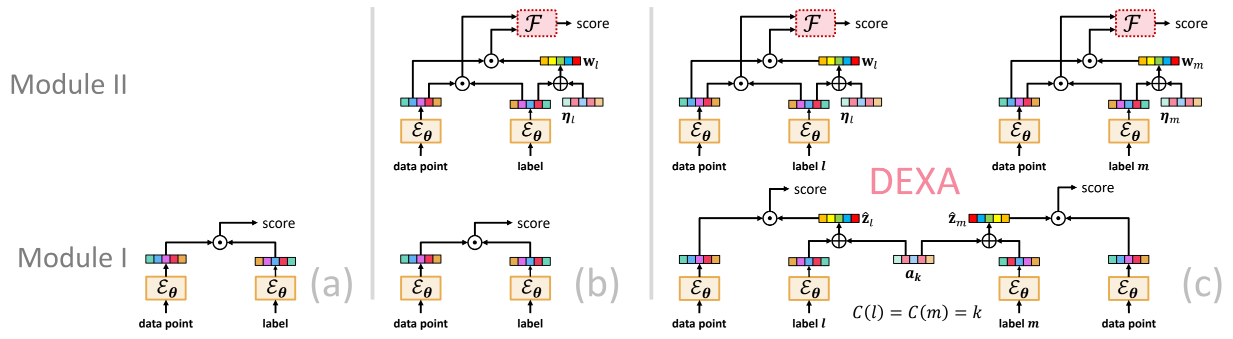

Figure 1: A depiction of the architectural modification introduced by DEXA. The 3 vertical panels denote 3 distinct training styles. (a) Encoder-only Models e.g. [31, 56]– these models do not incorporate a classifier and make predictions based on encoder embeddings alone. Training consists of a single module. (b) Encoder + Classifier Models *e.g.,* [14, 61] – these incorporate an encoder and a classifier. Training consists of two modules. The module I trains the encoder alone using a surrogate task, say using Siamese training itself whereas module II incorporates classifiers by freezing the encoder and perturbing label embeddings with correction terms η_l, one per label. (c) DEXA's model – this incorporates shared auxiliary vectors in module I itself. Notice that the two modules in DEXA are more similar to each other than the two modules for (b). This makes transitioning across the two models a smoother process and reduces distortions in encoder training. Training proceeds in two modules with the first module using shared auxiliary vectors while the second module give individual correction terms to each label.

overwhelmingly large majority of the labels are *tail* labels which occur infrequently during training making it challenging to train accurate classifier architectures. Training is made even more challenging by the fact that applications with millions of labels also usually offer training sets with millions of training data points. Training on all data point-label pairs is usually infeasible, making some form of *negative sampling* necessary [18, 18, 34, 46, 47, 56].

Semantic Gap in XC: XC methods also face application-specific challenges. In contemporary XC applications, it is common to expect both data points and labels to be endowed with textual descriptions. A straightforward strategy in this case would be to use an encoder to embed both data points and labels in a shared space and perform training such that related data point-label pairs are embedded close-by. Then at inference time, the nearest neighbors of the embedding of the test data point could be used to predict its related labels. This strategy is indeed used in Siamese training methods [13, 56]. However, this approach is expected to suffer if the textual descriptions are not descriptive enough which makes it challenging for an encoder to bring related data points and labels close to each other in the embedding space. We refer to this as a *semantic gap* in the label description. Semantic gaps are especially common in short-text applications where label descriptions are extremely short. However, all XC applications present some degree of semantic gap. Table 10 presents an example where the Wikipedia document titled "Constitutional reforms of Julius Caesar" where the textual description may not be sufficient to predict related pages such as 'Acta Senatus'[1]. To overcome this, XC methods introduce a high-capacity classifier [9, 24, 36] such as 1-vs-all models [8, 9, 13, 49] where each label is endowed with a linear classifier that acts on the embedding of a data point. It is notable that the use of alternate embedding-only architectures such as using a separate encoder to encode labels (dual encoders) cannot by themselves address the semantic gap completely as label representations in a dual encoder still depend on label text alone.

[1]reform of recording and issuing minutes of discussions and decisions of the Roman Senate

Challenges in XC Training: The introduction of classifier architectures such as 1-vs-all increases model capacity but also makes joint training challenging, especially on a single GPU. Modular training strategies are a popular workaround [8, 13, 15, 61] wherein the encoder is first trained by itself on a surrogate task and frozen. Subsequently, the classifier architecture is trained with respect to this frozen encoder and optionally, a few epochs of joint training are performed. This approach, along with other strategies such as negative mining, make training fairly scalable and XC models can be trained on tasks with several millions of labels on a single GPU. However, this paper argues that this strategy introduces distortion into encoder training and offers suboptimal performance.

Contributions: This paper makes six key contributions:

(1) The paper points out that the semantic gap in XC applications, especially short-text applications, can lead to distorted training in the first modular step where the encoder is trained solo.
(2) The paper introduces a light-weight alternative DEXA that augments the encoder with auxiliary parameters so that label representations are not constrained by label text alone.
(3) DEXA can be readily incorporated into existing XC architectures with minimal modifications (see Table 5).
(4) The paper outlines training strategies with the DEXA modification and shows how classifier architectures can still be utilized. Implementations are presented that offer training on datasets with upto 40 million labels on a single GPU.
(5) DEXA offers provably superior performance than traditional Siamese training in certain realizable settings.
(6) On benchmark and proprietary datasets, DEXA's embeddings offer respectively 6% and 15% more accurate performance than embeddings offered by existing XC methods. Using DEXA entails minimal overheads on the training time and no overhead on model size (see Table 8).

2 RELATED WORKS

Early XC Works – Fixed Features: XC models have undergone radical transformation in recent years, mostly due to the successful inclusion of deep encoder architectures [54]. The earliest XC models were reliant on sparse bag-of-words features and later graduated to using pre-trained dense features. The focus of model training was solely on training an accurate and scalable classifier architecture [1–4, 6, 21, 21–23, 27, 35, 39–43, 52, 57].

Task-specific Features – Siamese Training: Contemporary XC models mostly use deep learnt, task specific features. As mentioned in Section 1, having access to textual descriptors for both data points and labels allows us to train encoders that embed both into a shared embedding space such that related data points and labels are embedded in close proximity of each other. A number of works have explored this direction. For instances, TwinBERT [31] takes a large, pre-trained transformer model, fine-tunes it on ad retrieval tasks and then distils the resulting model. ANCE [30, 56] and RocketQA [45] use similar Siamese-style architectures but focus on developing effective negative sampling techniques to accelerate training. Note that these works do not utilize classifier architectures and rely on encoder embeddings alone to perform prediction.

Encoder+Classifier Models: Several works have established that the inclusion of classifier architectures into the model in addition to the encoder gives sustained and significant boosts to model accuracy. This boost is especially large on short-text applications indicating a large semantic gap which does not allow encoders to offer good performance by themselves. A large variety of encoder architectures (bag-of-embeddings, CNNs, LSTMs and transformers) and classifier architectures (1-vs-all, trees) have been used. Leading deep XC methods include XT [55], AttentionXML [60], APLC-XLNet [59], Astec [15], XR-Transformer [61], SiameseXML [13], LightXML [24], CascadeML [28] and ELIAS [19]. Deep architectures have also been used to incorporate label metadata such as label-graphs [38, 49] or multi-modal label data [37].

Model Training: When training with millions of training data points and millions of labels, training on all data point-label pairs becomes infeasible irrespective of the loss function being used or whether using an encoder-only model or an encoder+classifier architecture. Since the number of relevant labels per data point is usually small, the focus shifts on finding for each data point, a small set of labels that are irrelevant for that data point but which included in training, give most useful signals to the model. A variety of such *hard negative mining* models have been explored in recent years including simple techniques such as random sampling [34], in-batch sampling [10, 13, 17, 18, 20] and more involved methods that use nearest neighbor data structures or maximum inner product search (MIPS) data structures [32] to identify hard-negatives for a data point [13, 56, 60, 61]. Furthermore, the use of modular training strategies is popular with a surrogate task being used to pre-train the encoder separately followed by classifier training with a now-fixed encoder [9, 13, 15, 25, 60, 61]. Learning the shared encoder in a Siamese fashion can provide an accurate representation for data-scarce tail labels. Modular training can also be efficient in terms of compute (as only encoder or classifier need to be updated in an iteration and not both) as well as memory (at any point of time, optimizers such as Adam need to store first and second order moments for either the encoder or classifier but not both).

3 PROBLEM SETTING

Notation: Let L be the total number of labels in the application. Note that the label set remains same across training and testing. Let $\mathbf{x}_i, \mathbf{z}_l \in \mathcal{X}$ be the textual descriptions of the data point i and label l respectively. The training set is comprised of N data points and L labels as $\mathcal{D} := \{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, \{\mathbf{z}_l\}_{l=1}^L\}$. For each data point $i \in [N]$, its ground truth label vector is $\mathbf{y}_i \in \{-1, +1\}^L$, where $y_{il} = +1$ if label l is relevant to the data point i and otherwise $y_{il} = -1$.

Model Architecture: We assume that an encoder $\mathcal{E}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{S}^{D-1}$ is used to embed data points and labels with $\boldsymbol{\theta}$ being trainable parameters of the encoder. $\mathcal{S}^{D-1}$ denotes the D-dimensional unit sphere, *i.e.,* the encoder provides D-dimensional, unit norm embeddings. For simplicity, assume an 1-vs-all-style classifier architecture as $\mathbf{W} \overset{\text{def}}{=} \{\mathbf{w}_l\}_{l \in [L]}$ where $\mathbf{w}_l$ is the classifier for label l.

Making Joint Training Scalable: A straightforward way to jointly train both the encoder $\boldsymbol{\theta}$ and the classifier $\mathbf{W}$ architectures could have been to utilize a loss function of the following kind

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

which trains a data point i with respect to every (relevant, irrelevant) label pair, encouraging the classifier score for the relevant label to be higher than that of the irrelevant label by a margin γ. Usually the number of relevant labels per data point is around $O(\log L)$ [5] which means that this summation contains $\Omega(NL \log L)$ terms which is prohibitive when training with millions of data points and millions of labels. In an effort to reduce this time complexity, two steps are usually taken that are discussed below.

Hard Negative Mining: Instead of training a data point with respect to all its irrelevant labels, most of whom may not even provide useful signals to the model, training is only done with respect to a subset of $O(\log L)$ irrelevant labels which are expected to provide the most informative signals to model training. For example, ANCE [56] chooses to train on the hardest-to-predict irrelevant labels. The loss function used would thus look like

$$\min_{\boldsymbol{\theta}, \mathbf{W}} \mathcal{L}(\boldsymbol{\theta}, \mathbf{W}) = \sum_{i=1}^{N} \sum_{\substack{l: y_{il}=+1 \\ m \in \hat{\mathcal{N}}_i}} [\mathbf{w}_m^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{w}_l^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) + \gamma]_+, \quad (2)$$

where $\hat{\mathcal{N}}_i$ is the set of hard negatives for data point i with $\left|\hat{\mathcal{N}}_i\right| \approx O(\log L)$. Note that this restricted form of training reduces the number of terms in the summation to $O\left(N \log^2 L\right)$ making it scalable since usually $|\{l : y_{il} = +1\}| \approx O(\log L)$, *i.e.,* data points usually have around $O(\log L)$ relevant labels [5].

Modular Training: The second step taken to make learning scalable is to split training into multiple modules [9, 15, 19, 36, 61]. Although the exact splitting strategy varies from method to method, a typical strategy [13] is to first assume that the label classifier can be approximated by the label embedding itself, *i.e.,*

$$\mathbf{w}_l \approx \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$$

and train the encoder all by itself as:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{\substack{l: y_{il}=+1 \\ m \in \hat{\mathcal{N}}_i}} [\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_m)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) + \gamma]_+, \tag{3}$$

where $\hat{\mathcal{N}}_i$ is the set of hard negatives for data point i. Note that the above expression has no classifiers $\mathbf{W}$. Once the encoder has been learnt this way, it is frozen and the label classifiers are learnt by introducing a correction term and modelling the classifiers as

$$\mathbf{w}_l = \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) + \boldsymbol{\eta}_l$$

The correction terms can then be learnt as

$$\min_{\{\boldsymbol{\eta}_l\}} \mathcal{L}(\{\boldsymbol{\eta}_l\}) = \sum_{i=1}^{N} \sum_{\substack{l: y_{il}=+1 \\ m \in \hat{\mathcal{N}}_i}} [(\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_m) + \boldsymbol{\eta}_m)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - (\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) + \boldsymbol{\eta}_l)^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) + \gamma]_+, \tag{4}$$

In practice, instead of having explicit correction terms, the label classifiers are instead initialized to the label embeddings i.e.

$$\mathbf{w}_l \overset{\text{init}}{\longleftarrow} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$$

and then $\mathbf{w}_l, l \in [L]$ are directly fine tuned using (4).

4 DEXA: DEEP ENCODERS WITH AUXILIARY PARAMETERS

We notice that the assumption $\mathbf{w}_l \approx \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$ inherent to the above approach may not hold uniformly well for all labels, and be difficult to justify in applications where textual descriptions are scant such as short-text applications. In such cases, the first step of modular training (3) may learn a distorted encoder as the model parameters $\boldsymbol{\theta}$ twist themselves in an effort to ensure that $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) \longrightarrow \mathbf{w}_l$. This distorted encoder may continue to offer suboptimal performance even when corrections are introduced in (4). It is possible to somewhat remedy the situation by amending (4) to jointly fine-tune $\boldsymbol{\theta}$ along with learning $\{\boldsymbol{\eta}_l\}$. However, joint fine-tuning is usually done for few epochs otherwise the benefits of modular training are entirely lost. This may not be able to fully undo the encoder distortions introduced during (3) training.

Intuition Behind DEXA: In situations with large semantic gap, the simplest solution is to offer labels correction terms during encoder training itself. However, to avoid the steep computational cost of introducing L correction terms in (3), DEXA postulates that "related" labels may have similar correction terms. Specifically, if label l and m are "related", then

$$\mathbf{w}_l - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) \approx \mathbf{w}_m - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_m)$$

Lemma 1 below shows that when this condition is met, DEXA offers provably better encoder training than pure Siamese-style training methods. Note that the above postulate is distinct from the postulate $\mathbf{w}_l \approx \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l)$ made (implicitly) by existing methods. DEXA exploits this intuition by introducing $K \ll L$ auxiliary variables in encoder training. DEXA's training procedure is described below.

Model Architecture: DEXA uses a DistilBERT base [50] encoder as $\mathcal{E}_{\boldsymbol{\theta}}$ to embed data points and labels. Additionally, during training, DEXA also uses K auxiliary vectors $\mathbf{A} \overset{\text{def}}{=} \{\mathbf{a}_k\}_{k\in[K]}$ where $\mathbf{a}_k \in \mathbb{R}^D$ to train the encoder. We note that we do not need to explicitly store auxiliary vectors once training is completed and that they do not add to DEXA's overall model size. DEXA also uses a 1-vs-all-style classifier architecture as $\mathbf{W} \overset{\text{def}}{=} \{\mathbf{w}_l\}_{l\in[L]}$ where $\mathbf{w}_l$ is the classifier for label l.

Encoder Training – Module I: First, DEXA creates $K \ll L$ disjoint label clusters, say $C_1, C_2, \ldots, C_l$, *i.e.,* $C_i \cap C_j = \emptyset$ and $\bigcup_{k\in K} C_k = [L]$. Let $C : [L] \to [K]$ denote the cluster assignment operator, *i.e.,* for any $l \in [L], C(l) \in [K]$ denotes its cluster. To create its clusters, DEXA uses a pre-trained model, say with parameters $\boldsymbol{\theta}^0$ using which pre-trained label embeddings are obtained $\mathcal{E}_{\boldsymbol{\theta}^0}(\mathbf{z}_l), l \in [L]$. These embeddings are then subject to hierarchical balanced k-means clustering [13] to create K balanced label clusters. Next, an auxiliary vector $\mathbf{a}_k \in \mathbb{R}^D$ is assigned to cluster C_k. Note that this requires additionally storing K auxiliary vectors. Then the label classifier for label $l \in [L]$ is approximated as

$$\mathbf{w}_l = \mathfrak{N}(\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) + \mathbf{a}_{C(l)}),$$

where $\mathfrak{N} : \mathbb{R}^D \to \mathcal{S}^{D-1}$ is the normalization operator defined as $\mathfrak{N}(\mathbf{v}) \overset{\text{def}}{=} \mathbf{v}/\|\mathbf{v}\|_2$. For a data point $i \in [N]$, let $\mathcal{N}_i \subset [L]$ denote the hard negative labels offered by some hard negative mining strategy. Note that $y_{il} = -1$ for all $l \in \mathcal{N}_i$. DEXA uses the in-batch-style negative mining strategy proposed in [14] as it had less memory overheads compared to other methods yet offered fast convergence. Encoder training is then done using the following loss function:

$$\min_{\{\boldsymbol{\eta}_l\}} \mathcal{L}(\{\boldsymbol{\eta}_l\}) = \sum_{i=1}^{N} \sum_{\substack{l: y_{il}=+1 \\ m \in \hat{\mathcal{N}}_i}} [\mathfrak{N}(\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_m) + \mathbf{a}_{C(k)})^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathfrak{N}(\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) + \mathbf{a}_{C(l)})^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) + \gamma]_+, \tag{5}$$

Note that the above formulation affords (shared) correction terms to all labels. Also, only the hard negative labels and relevant labels of a data point participate in training. To further accelerate training, while creating a mini-batch, a single positive label was randomly sampled for each data point in the mini-batch. Note that this choice continues to offer an unbiased estimate of the above loss function. Experiments were also conducted with dynamic cluster assignment where the clusters were regularly refreshed during Module I training using the recent-most label embeddings computed using the encoder $\mathcal{E}_\theta$ that was getting trained in parallel. This dynamic clustering variant led to only marginal gains in performance.

Classifier Training – Module II: Once encoder training is complete, the encoder is frozen and *augmented embedding* of each label is computed as

$$\hat{\mathbf{z}}_l \overset{\text{def}}{=} \mathfrak{N}(\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) + \mathbf{a}_{C(l)})$$

The augmented label embeddings are preserved but the auxiliary vectors $\mathbf{a}_k, k \in [K]$ are now discarded. Next, the label classifiers are initialized to the augmented label embeddings, *i.e.,*

$$\mathbf{w}_l \overset{\text{init}}{\longleftarrow} \hat{\mathbf{z}}_l$$

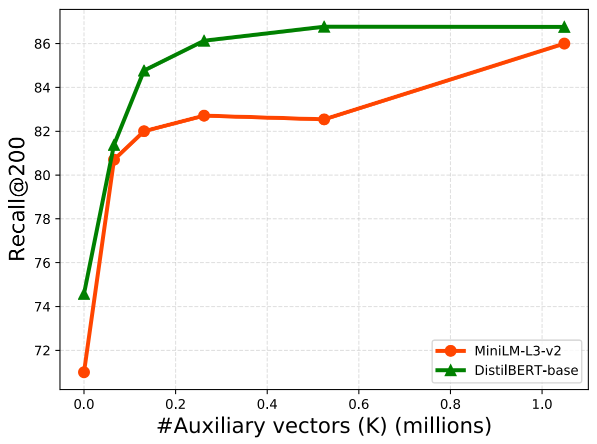

Figure 2: Performance of DEXA with number of auxiliary vectors on the proprietary SponsoredSearch-40M dataset. DEXA yielded 7-15% gains over NGAME embeddings. Please note that $K = 0$ in the figure corresponds to NGAME. The 3-layer MiniLM-L3-v2 model could perform on par with a 6 layer DistilBERT model when both are augmented with $K \approx L/40$ auxiliary vectors

.

and the classifiers are learnt using

$$\min_{\{\mathbf{w}_l\}} \mathcal{L}(\{\mathbf{w}_l\}) = \sum_{i=1}^{N} \sum_{\substack{l: y_{il}=+1 \\ m \in \hat{\mathcal{N}}_i}} [\mathbf{w}_m^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{w}_l^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_i) + \gamma]_+, \quad (6)$$

It is possible to modify the above optimization problem to jointly fine-tune the encoder parameters $\boldsymbol{\theta}$ as well but the default version of DEXA does not do this in favor of scalability.

Inference Pipeline: The training process proceeds in two modules and is described below in detail. In summary, module trains the encoder $\mathcal{E}_{\boldsymbol{\theta}}$ using the auxiliary vectors after which the classifiers $\mathbf{W}$ are trained. Afterward, a maximum inner product search (MIPS) [32] data structure is created over the set of label classifiers i.e., $\{\mathbf{w}_l\}_{l \in [L]}$. For a test data point $\mathbf{x}_t$, its encoder embedding $\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_t)$ is sent as a query into the MIPS data structure which returns a specified number of labels $l \in [L]$ whose classifier score $\mathbf{w}_l^\top \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}_t)$ is the highest. The size of this shortlist is typically set to $100 \approx O(\log L)$. Similar to existing methods [13–15, 22], DEXA combines classifier scores with label similarity scores that are computed separately. Labels are sorted according to this combined score and predictions are made (see supplementary material for details). Moreover, DEXA has no overheads with respect to model size and inference time complexity (see supplementary material for details). In particular, DEXA offers test time predictions in time $O(B + D \log L)$ where $B = |\boldsymbol{\theta}|$ is size of the encoder model and D is the embedding dimensionality of the encoder. Table 8 shows that the training time complexity overheads of DEXA are minimal.

Head Label Partitioning: Labels in XC applications are often designated as being either *head*, *torso*, or *tail* depending on how frequently do they occur in the training set, *i.e.,* to how many training data points is a label relevant. Head labels are relevant to a large number of data points whereas tail labels are often relevant to less than 10 training points [5]. This implies that head labels stand to gain the most from correction terms since they have enough training data to utilize the additional capacity. For this reason, a variant of DEXA was developed that takes $\hat{K}$ of the most popular head labels, and assigned them a cluster of their own. The rest of the $L - \hat{K}$ labels are then partitioned into $K - \hat{K}$ clusters. This strategy was found to offer mild improvements in performance. DEXA decides $\hat{K}$ by counting the number of labels that are relevant to at least 50 training points.

5 THEORETICAL ANALYSIS

We theoretically establish here DEXA's ability to reduce the effect of semantic gap on encoder training, on a realizable model. We note that using simple extensions of existing results on generalizability of encoder-based XC architectures [13], it can be shown that the model learnt by DEXA after encoder training (Module I) enjoys generalization bounds that entirely independent of the number of labels L in the application. Instead, they are dependent only on the size of the encoder and K, the number of auxiliary vectors used in Module I training.

5.1 Realizable Model

Consider a setting where the N data points and the L labels are described using P-dimensional vectors i.e., $\mathbf{x}_i, \mathbf{z}_l \in \mathbb{R}^P$. We consider a linear encoder parameterized by a matrix $\mathbf{E} \in \mathbb{R}^{D \times P}$ as $\mathcal{E}(\mathbf{x}) = \mathbf{E}\mathbf{x} \in \mathbb{R}^D$. Note that we do not enforce the embeddings to be unit norm to simplify the calculations. Also, assume a regression setting where $y_{il} \in \mathbb{R}$. Furthermore, we consider the *realizable* setting where there are some (unknown) encoder and classifier parameters which have generated the data i.e., for some $\mathbf{E}^* \in \mathbb{R}^{D \times P}$ and $\mathbf{w}_l^* \in \mathbb{R}^D, l \in [L]$, we have

$$y_{il} \overset{\text{def}}{=} (\mathbf{w}_l^*)^\top \mathbf{E}^* \mathbf{x}_i$$

The semantic gap in this setting can be quantified as follows:

$$\boldsymbol{\Delta}_l \overset{\text{def}}{=} \mathbf{w}_l^* - \mathbf{E}^* \mathbf{z}_l$$

Let us now try to perform encoder learning using classical Siamese strategy and DEXA using the squared loss.

Siamese Training: Adapted to this setting, (3) would use the following loss function

$$\mathcal{L}(\mathbf{E}) = \frac{1}{NL} \sum_{i,l} (\mathbf{z}_l^\top \mathbf{E}^\top \mathbf{E}\mathbf{x}_i - y_{il})^2 \quad (7)$$

DEXA Training: Adapted to this setting, Module I of DEXA would cluster the labels into K clusters and use the following loss function along with auxiliary variables $\mathbf{A} \overset{\text{def}}{=} \{\mathbf{a}_k\}_{k \in [K]}$

$$\mathcal{L}(\mathbf{E}, \mathbf{A}) = \frac{1}{NL} \sum_{i,l} ((\mathbf{E}\mathbf{z}_l + \mathbf{a}_{C(l)})^\top \mathbf{E}\mathbf{x}_i - y_{il})^2 \quad (8)$$

where $C(l)$ gives us the cluster of label l.

Comparison: The minimizers of both the above optimization problems are not available in closed form despite the simplified setting as these continue to be non-convex problems. To get around this issue, we look at the norm of the gradient of the two objective functions at the optimal encoder i.e. we compare $\|\nabla_E \mathcal{L}(\mathbf{E}^*)\|_2$ with $\|\nabla_E \mathcal{L}(\mathbf{E}^*, \mathbf{A})\|_2$. A larger encoder gradient at $\mathbf{E}^*$ suggests that the optimal encoder for that objective function lies farther from $\mathbf{E}^*$ which will allow us to quantify the suitability of the two objective functions. Let us define some handy notation. For any label cluster $k \in [K]$, let $\overline{\mathbf{\Delta}}_k$ denote the average semantic gap in cluster C_k, *i.e.,*

$$\overline{\mathbf{\Delta}}_k \overset{\text{def}}{=} \frac{1}{|C_k|} \sum_{l \in C_k} \mathbf{\Delta}_l$$

Also let σ_k^2 denote the intra-cluster variance in semantic gap i.e.,

$$\sigma_k^2 \overset{\text{def}}{=} \sum_{l \in C_k} \|\mathbf{\Delta}_l - \overline{\mathbf{\Delta}}_k\|_2^2$$

Then we have the following result with us

Lemma 1. *Assuming* $\|\mathbf{x}_i\|_2, \|\mathbf{z}_l\|_2, \|\mathbf{w}_l^*\|_2 \leq 1$ *to avoid notational clutter, we have*

$$\|\nabla_E \mathcal{L}(\mathbf{E}^*)\|_2 \leq 2\|\mathbf{E}^*\|_2^2 \sqrt{\frac{1}{L} \sum_{l \in [L]} \|\mathbf{\Delta}_l\|_2^2}$$

$$\|\nabla_E \mathcal{L}(\mathbf{E}^*, \mathbf{D}^*)\|_2 \leq 4\|\mathbf{E}^*\|_2^2 \sqrt{\frac{1}{L} \sum_{k \in [K]} \sigma_k^2},$$

where $\mathbf{D}^* = [-\overline{\mathbf{\Delta}}_l, \ldots, -\overline{\mathbf{\Delta}}_K] \in \mathbb{R}^{D \times K}$ *and* $\|\mathbf{E}^*\|_2$ *is the spectral norm of the matrix* $\mathbf{E}^*$.

The above result shows that the DEXA approach offers a far smaller encoder gradient, indicating a more faithful recovery of the true encoder parameters, if

$$\sigma_k^2 \ll \sum_{l \in C_k} \|\mathbf{\Delta}_l\|_2^2$$

i.e., if

$$\sum_{l \in C_k} \|\mathbf{\Delta}_l - \overline{\mathbf{\Delta}}_k\|_2^2 \ll \sum_{l \in C_k} \|\mathbf{\Delta}_l\|_2^2$$

Thus, even if the individual label-wise semantic gaps $\mathbf{\Delta}_l$ are large in a cluster, DEXA offers faithful encoder recovery so long that those (large) semantic gaps are similar to each other. Recall that this is exactly the postulate made in Section 4 which posits good performance for DEXA if for labels l, m in the same cluster, we have

$$\mathbf{w}_l - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_l) \approx \mathbf{w}_m - \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{z}_m)$$

Due to lack of space, the proof of this result is presented in the supplementary material.

Generalization Bound: DEXA's frugal strategy allows it to offer crisp generalization bounds. We present an informal statement here with a detailed discussion in the supplementary material.

Lemma 2 (Excess Risk Bound – Informal). *Suppose DEXA is used with an encoder with parameters* $\boldsymbol{\theta}$ *and offering D-dimensional embeddings. For a confidence parameter* $\delta \in (0, 1]$, *suppose the encoder offers the following excess risk bound with probability* $1 - \delta$,

$$\ell(\boldsymbol{\theta}) \leq \hat{\ell}_N(\boldsymbol{\theta}) + \epsilon(N) + \sqrt{\frac{\ln \frac{1}{\delta}}{N}}$$

where $\epsilon(N)$ *captures the dependence of the excess risk on the encoder parameter characteristics and* $\ell, \hat{\ell}_N$ *respectively denote the population risk and empirical risk on N i.i.d. training points. Then DEXA with K auxiliary vectors collected in the matrix* $\mathbf{A} \in \mathbb{R}^{D \times K}$ *offers an excess risk bound of the form*

$$\ell(\boldsymbol{\theta}, \mathbf{A}) \leq \hat{\ell}_N(\boldsymbol{\theta}, \mathbf{A}) + \epsilon(N) + \sqrt{\frac{\ln \frac{1}{\delta}}{N}} + \frac{\Delta \ln N}{\sqrt{N}},$$

where $\Delta \overset{\text{def}}{=} O\left(\ln(DK)\sqrt{R_1 R_\infty}\right)$ *and* $R_1 \overset{\text{def}}{=} \|\mathbf{A}\|_{1,1}, R_\infty \overset{\text{def}}{=} \|\mathbf{A}\|_{1,\infty}$.

Table 1: Dataset Statistics for benchmark datasets (publicly available on The Extreme Classification Repository [5]) and the ORCAS dataset [12]

Dataset	# Train Pts N	# Labels L	# Test Pts N'	Avg. # data pts per label	Avg. # labels per data pt
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
ORCAS-800K	7,371,396	799,953	2,551,398	16.12	1.75
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

Table 2: Results on short-text benchmark datasets with embeddings trained using different algorithms. See the supplementary material for full results, definitions of P@k, PSP@k.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
LF-AmazonTitles-1.3M						
DEXA	**51.92**	**44.01**	**38.86**	32.96	**35.86**	**37.31**
NGAME	45.82	39.94	35.48	**33.03**	35.63	36.8
SiameseXML	43.80	38.60	34.94	21.64	25.89	28.48
ECLARE	38.80	34.54	31.51	22.25	26.13	28.32
DECAF	41.07	36.56	33.30	17.43	20.85	23.01
ANCE	46.44	41.48	37.59	31.91	35.31	37.25
DPR	44.64	39.05	34.83	32.62	35.37	36.72
LF-AmazonTitles-131K						
DEXA	**44.76**	**29.72**	**21.18**	**39.29**	**44.58**	**49.50**
NGAME	42.61	28.86	20.69	38.27	43.75	48.71
SiameseXML	38.13	26.11	19.17	35.09	39.97	45.14
ECLARE	33.42	24.44	18.35	30.4	36.68	42.33
DECAF	33.51	23.08	17.05	27.93	33.13	38.33
ANCE	42.67	29.05	20.98	38.16	43.78	49.03
DPR	41.85	28.71	20.88	38.17	43.93	49.45
BM25	18.10	11.69	8.51	17.94	18.53	20.73
ColBERT	12.87	9.99	7.18	14.19	14.57	16.16
GraphFormers	20.84	13.57	10.06	20.82	21.85	24.93
RocketQA	42.75	29.22	20.98	39.97	44.50	49.21

6 EXPERIMENTS

Datasets: This paper considers several benchmark datasets publicly available at the Extreme Classification Repository [5]. The label-metadata is available in the form of label-text for these datasets. In particular, LF-AmazonTitles-131K, LF-AmazonTitles-1.3M involve predicting other relevant products based on product title, whereas both product title and description are available for LF-Amazon-131K dataset. Additionally, LF-Wikipedia-500K and LF-WikiSeeAlso-320K involve predicting relevant Wikipedia categories and similar

Table 3: Results on full-text benchmark datasets with embeddings trained using different algorithms. See the supplementary material for full results, definitions of P@k, PSP@k.

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
			LF-Wikipedia-500K			
DEXA	**79.99**	**57.08**	**42.52**	50.15	**56.24**	**57.68**
NGAME	77.92	54.87	40.95	**50.99**	56.02	57.33
SiameseXML	50.33	32.81	24.86	33.17	31.61	32.51
ECLARE	63.41	43.20	33.44	29.64	33.32	36.23
DECAF	71.64	52.24	40.92	33.02	40.27	44.35
ANCE	63.33	43.35	33.12	34.95	37.51	39.71
DPR	65.23	45.85	35.23	45.79	47.83	49.90
			LF-WikiSeeAlso-320K			
DEXA	**46.57**	**29.92**	**22.26**	**32.38**	**35.34**	**38.27**
NGAME	43.58	28.01	20.86	30.59	33.29	36.03
SiamseXML	40.70	27.16	20.74	29.05	32.39	35.67
ECLARE	35.21	24.36	18.89	24.48	28.89	31.62
DECAF	37.66	24.67	18.59	24.83	27.64	30.32
ANCE	44.35	29.15	21.99	30.32	33.80	37.15
DPR	41.66	27.16	20.66	30.32	32.97	36.25
			LF-Amazon-131K			
DEXA	**46.64**	**30.93**	**22.06**	**38.83**	**44.98**	**50.38**
NGAME	45.35	29.89	21.35	38.53	44.08	49.32
SiameseXML	41.97	28.59	21.03	37.28	42.95	48.90
ECLARE	34.80	25.60	19.41	30.53	37.78	44.29
DECAF	39.04	26.14	19.03	31.92	37.31	42.67
ANCE	44.87	30.31	21.89	37.94	44.36	50.12
DPR	43.30	29.74	21.90	38.30	44.98	51.52

Table 4: Results on the ORCAS-800K dataset. +AP indicates whether the Auxiliary Parameters were used alongside the backbone architecture.

Architecture	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	R@10
DistilBERT-Base	72.47	38.87	26.60	58.70	70.26	76.68	87.84
DistilBERT-Base + AP	**74.98**	**40.46**	**27.58**	**60.27**	**72.71**	**79.13**	**89.77**

Wikipedia articles, respectively. Please note that the paper considers short-text (LF-AmazonTitles-131K and LF-AmazonTitles-1.3M) as well as full-text (LF-Amazon-131K, LF-WikiSeeAlso-320K, and LF-Wikipedia-500K) benchmark datasets. Results are also reported on the ORCAS-800K dataset where the task is to predict URLs given a query. The dataset was created from click logs for a document retrieval task [11] (see supplementary material for details on dataset creation). Please refer to Table 1 for data statistics. Finally, results have also been reported on SponsoredSearch-40M dataset created by mining search engine logs from a popular search engine. This is an application specific dataset that tries to match user queries on a search engine to advertiser bid keywords.

Baselines: Improving the quality of label embeddings is the key contribution of the DEXA algorithm. The embeddings learned using DEXA are compared against the embeddings learned using encoder-only models. These include XC methods such as NGAME [14], SiameseXML [13], DECAF [36] and ECLARE [38]. Results are also reported for leading sparse and dense retrieval (DR) methods including BM25, DPR [26], ColBERT [29], GraphFormers [58], ANCE [56], and RocketQA [45]. Unfortunately, some methods could only scale to small datasets. Although not the focus of the paper, one could learn extreme classifiers on top of DEXA's embeddings either using

Table 5: Ablation experiments with different architectures. +AP indicates whether the Auxiliary Parameters were used alongside the backbone architecture or not.

Architecture	+AP	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
			LF-AmazonTitles-131K				
Bag-of-embedding	✗	29.45	20.35	15.11	26.69	30.76	35.41
	✓	36.72	24.03	17.19	30.29	34.30	38.47
Bi-GRU	✗	36.90	24.68	17.62	32.30	36.49	40.59
	✓	37.42	24.44	17.36	30.93	34.82	38.73
MiniLM-L3	✗	35.80	24.24	17.68	32.05	36.40	41.22
	✓	39.08	25.55	18.34	32.81	37.03	41.59
MiniLM-L6	✗	39.12	26.40	19.07	35.00	39.70	44.53
	✓	41.68	27.70	19.91	36.58	41.33	46.29
DistilBERT-Base	✗	43.17	28.99	20.73	38.74	43.94	48.81
	✓	44.48	29.50	21.08	39.26	44.43	49.36
BERT-Base	✗	43.49	29.01	20.75	39.07	44.03	48.92
	✓	45.44	29.76	20.99	39.03	43.99	48.43
			LF-AmazonTitles-1.3M				
Bag-of-embedding	✗	32.18	26.94	23.51	25.46	26.19	26.47
	✓	42.90	36.06	31.77	26.11	28.28	29.47
Bi-GRU	✗	36.01	30.18	26.20	26.66	27.53	27.81
	✓	43.38	36.23	31.81	27.50	29.21	30.08
MiniLM-L3	✗	35.85	30.01	26.11	27.92	28.79	29.08
	✓	39.64	31.94	27.42	27.59	28.63	29.01
MiniLM-L6	✗	38.72	32.80	28.70	29.71	30.97	31.42
	✓	43.95	36.28	31.59	30.42	31.90	32.55
DistilBERT-Base	✗	45.83	39.94	35.48	33.03	35.63	36.80
	✓	50.28	42.32	37.19	33.21	35.78	37.00
BERT-Base	✗	49.53	43.38	38.51	35.52	38.60	39.86
	✓	54.00	46.03	40.64	35.33	38.47	39.97

Table 6: Ablation experiments with different number of clusters on the LF-AmazonTitles-131K dataset. HLP refers to head label partitioning indicating whether the head labels were assigned separate auxiliary parameters or not.

#clusters	HLP	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
			LF-AmazonTitles-131K				
4,096	✗	43.48	28.97	20.70	38.72	43.88	48.71
8,192	✗	43.50	29.01	20.74	38.82	43.92	48.79
16,384	✗	43.78	29.07	20.77	38.84	43.99	48.80
32,768	✗	43.84	29.06	20.80	38.92	43.97	48.89
65,536	✗	44.48	29.50	21.08	39.26	44.43	49.36
32,768	✓	44.30	29.36	20.93	39.05	44.19	49.01
65,536	✓	44.76	29.72	21.18	39.29	44.58	49.50
			LF-AmazonTitles-1.3M				
4,096	✗	48.33	40.60	35.65	33.20	35.49	36.52
8,192	✗	48.66	40.82	35.82	33.18	35.48	36.51
16,384	✗	49.33	41.64	36.68	33.33	35.76	36.94
32,768	✗	49.54	41.72	36.67	33.30	35.74	36.89
65,536	✗	50.28	42.32	37.19	33.21	35.78	37.00
131,072	✗	51.47	43.23	37.97	32.74	35.46	36.76
65,536	✓	51.92	44.01	38.86	32.96	35.86	37.31
131,072	✓	52.51	44.49	39.27	32.82	35.86	37.38

Table 7: Ablation experiments with different clustering algorithms on the LF-AmazonTitles-131K dataset. Clustering could be performed over the label embedding ($\bar{Z}$) or label centroids ($\bar{X}$). DEXA deploys label partitioning (LP) to assign separate auxiliary parameters for head labels.

Method	space	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
k-means	$\bar{Z}$	43.28	28.98	20.77	38.41	43.82	48.79
Balanced k-means	$\bar{X}$	43.16	28.68	20.59	33.87	43.26	48.15
Balanced k-means	$\bar{Z}$	44.48	29.50	21.08	39.26	44.43	49.36
Balanced k-means + LP	$\bar{Z}$	44.76	29.72	21.18	39.29	44.58	49.50

Table 8: DEXA's implementation incurs minimal to no overheads in training time and model size as compared to NGAME's implementation when learning the encoder.

Metric	NGAME	DEXA	NGAME	DEXA
	LF-Wikipedia-500K		LF-AmazonTitles-1.3M	
Training time (hrs)	41.93	42.75	75.53	76.62
Model Size (GB)	1.82	1.82	4.71	4.71

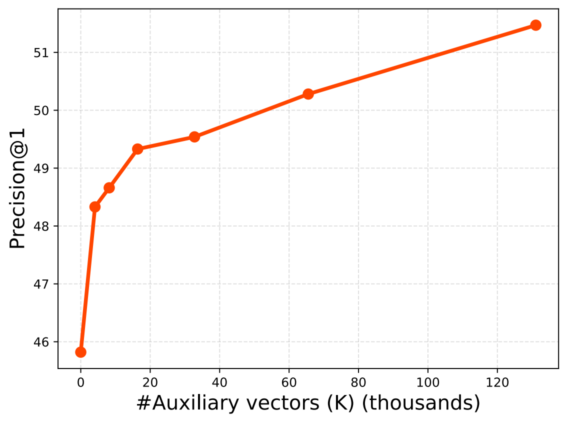

Figure 3: Performance of DEXA with number of Auxiliary Vectors on the LF-AmazonTitles-1.3M dataset. DEXA yielded 2.5-5.5% gains over NGAME embeddings which is equivalent to configuration $K = 0$ for DEXA. Please refer to Table 6 for more detailed results.

NGAME [14] or using a number of classical XC methods available to learn extreme classifiers over fixed features such as Slice [21] and Bonsai [27]. In particular, this paper reports results for the setting when the DEXA classifiers are learned using NGAME negative sampling. This setting is compared against leading deep XC techniques such as XR-Transformer [61], BERTXML [7], LightXML [24], AttentionXML [60] and MACH [33] that are endowed with extreme classifiers as well. Finally, the supplementary material includes results for classical XC methods including Parabel [41], DiSMEC [2] and Bonsai [27] for completeness. For SponsoredSearch-40M dataset, DEXA is primarily compared against NGAME which is one of the leading dense retrieval algorithms.

Evaluation Metrics: Results are reported on popular XC metrics such as precision@k (P@k, $k \in 1, 3, 5$) and their propensity-scored [22, 44, 51] variants precision@k (PSP@k, $k \in 1, 3, 5$). Moreover, results are also reported on vanilla and propensity-scored version of nDCG@k, *i.e.,* N@k ($k \in 1, 3, 5$) and PSN@k ($k \in 1, 3, 5$) respectively. Note that the propensity-scored variants put more emphasis on data-scarce tail labels. Please refer to The Extreme Classification Repository [5] for definitions of all these metrics. Different metrics might be important for different real-world applications. For *e.g.,* recall@k with large k reported for SponsoredSearch-40M dataset, is the primary metric for the application to match user queries to advertiser keywords. This is because the retrieval algorithm generally retrieves a large number of keywords which are progressively pruned in downstream relevance, click-prediction, and bidding stages.

Hyper-parameters: The most prominent of DEXA's hyper-parameters are the number of auxiliary vectors and the clustering algorithm. DEXA was found to be quite robust to these hyper-parameters as demonstrated in Table 6 and Table 7. DEXA was trained using NGAME negative sampling which has been demonstrated to yield state-of-the-art results in the XC setting. The Adam optimizer was used to learn the model parameters and its hyper-parameters including the number of epochs and learning rate. Please refer to Table 11 in the supplementary material for DEXA's hyper-parameters. The hyper-parameters of baseline methods were set as per the source paper wherever available and by fine-grained validation otherwise.

Evaluation on benchmark datasets: DEXA's embeddings resulted in more accurate predictions as compared to the embeddings learned using leading DR as well as XC methods as demonstrated in Table 2 and Table 3 for short-text and full-text datasets, respectively. In particular, DEXA's embeddings led to 2-32% more accurate predictions over leading DR methods including ANCE, DPR, ColBERT, GraphFormers, and RocketQA. Note that DR methods are specifically designed to learn an encoder-only model. Unfortunately, expensive methods such as ColBERT, GraphFormers and RocketQA could only scale to LF-AmazonTitles-131K dataset. Moreover, the embeddings learned using DEXA were also found to be 1.2–30% more accurate than the embeddings learned using leading XC methods including NGAME, SiameseXML, and DECAF. In particular, DEXA's embeddings could be up to 6% more accurate than the NGAME's embeddings on LF-AmazonTitles-1.3M dataset (see Fig. 3 for more details). Similar results were observed on the ORCAS-800K dataset (see Table 4) where adding auxiliary parameters led to a 2% gain over an encoder only model. Note that the same backbone architecture, *i.e.,* 6-layer DistilBERT-Base was used for both methods. Similar results were observed for other architectures including Bag-of-embeddings [15], Bi-GRU [60], 3-layer MiniLM [53], 6-layer MiniLM and BERT-Base [16] encoders demonstrating the generality of auxiliary parameters introduced by DEXA. Finally, the extreme classifiers trained over DEXA's embeddings could be up to 21% more accurate than leading deep XC methods including XR-Transformer, LightXML, AttentionXML, MACH and SiameseXML. Please see Table 13 and Table 14 in the supplementary material that includes results for methods endowed with extreme classifiers.

Evaluation on proprietary dataset: Fig. 2 shows the comparison of DEXA against NGAME which is one of the leading

Table 9: Light grey color is used to indicate wrong predictions. Example is taken from LF-AmazonTitles-1.3M dataset

Method	**Top 5 Predictions**
Document	**Fundamentals of Nuclear Pharmacy**
DEXA	Nuclear Medicine Technology: Procedures and Quick Reference Nuclear Medicine and PET/CT: Technology and Techniques, 7e Radiation Safety in Nuclear Medicine, Second Edition Practical Mathematics in Nuclear Medicine Technology Nuclear Medicine Instrumentation
NGAME	Basic Skills in Interpreting Laboratory Data, 4th Edition Ansel's Pharmaceutical Dosage Forms and Drug Delivery Systems Frequently Prescribed Medications: Drugs You Need to Know Pharmacy Practice And The Law Pharmaceutical Compounding and Dispensing

Table 10: Light grey color is used to indicate wrong predictions. Example is taken from LF-WikiSeeAlso-320K dataset

Method	**Top 5 Predictions**
Document	**Constitutional reforms of Julius Caesar**: The constitutional reforms of Julius Caesar were a series of laws pertaining to the Constitution of the Roman Republic enacted between 49 and 44 BC, during Caesar's dictatorship. Caesar died in 44 BC before the implications of his cons- -titutional actions could be realized. During his early career, Caesar had seen how chaotic and dysfunctional the Roman Republic had become. The republican machinery had broken down under the weight of imp- -erialism, the central government had become powerless, the pro- -vinces had been transformed into independent principalities under the absolute control of their governors, and the army had replaced the constitution as the means of accomplish
DEXA	Acta Senatus Centuria Roman Law Interrex Byzantine Senate
NGAME	Julius Caesar Assassination of Julius Caesar Caesarism Constitution of the Roman Republic Caesar's civil war

dense-retrieval algorithms in production for the Sponsored Search application. Please note that $K = 0$ in the figure corresponds to NGAME. DEXA with a smaller MiniLM-L3-v2 encoder with 17M parameters and augmented with just 200K auxiliary vectors was found to be 10% more accurate than NGAME with a DistilBERT encoder with 66M parameters. This indicates the effectiveness of the auxiliary vectors which when combined with a smaller encoder can lead to better results than state-of-the-art dense retrieval methods with a larger encoder. This ability of DEXA to enable a small encoder to perform better is significant for real-world applications which need to make billions of predictions per day under strict latency limits and can lead to significant COGS savings for such applications. The figure further shows how for a larger DistilBERT encoder, performance saturates faster than a smaller MiniLM-L3-v2 encoder as we increase the number of auxiliary vectors.

Ablations: Ablation experiments were conducted to analyze the impact of the clustering algorithm and the number of clusters. DEXA was found to be quite robust to the clustering algorithms as it yielded comparable accuracies with different strategies, *viz.,* clustering using standard k-means over label embeddings, balanced k-means over label embeddings and balanced k-means over label centroids. Note that label centroid for label l is computed as $\frac{\sum_{\{i|y_{il}=+1\}} \mathcal{E}_\theta(\mathbf{x}_i)}{\sum_{\{i|y_{il}=+1\}} 1}$. Please refer to Table 7 for results on the LF-AmazonTitles-131K dataset. Additionally, Fig. 3 demonstrates that introducing merely 2^{12} (or 4,096) clusters(and corresponding auxiliary parameters) led to a 2.5% gain in P@1 over NGAME which doesn't make use of auxiliary parameters indicating the impact of auxiliary parameters while training deep encoders. Additionally, increasing the number of clusters from 2^{12} to 2^{16} led to a 1% and 2% increase on LF-AmazonTitles-131K and LF-AmazonTitles-1.3M datasets, respectively. Moreover, introducing head label partitioning (HLP) yielded a further 1% gain over the vanilla variant. Please see Table 6 for detailed results. Increasing the number of clusters K can lead to gains in accuracy however that comes at the cost of memory overhead. In particular, introducing K clusters incurs $\mathcal{O}(KD)$ memory overhead where D is the embedding dimension. See supplementary material for detailed analysis.

Qualitative Analysis: Table 9 shows the top 5 predictions of DEXA and NGAME for the test document "Fundamentals of Nuclear Pharmacy" in the LF-AmazonTitles-1.3M dataset. DEXA was able to predict all of the top 5 slots correctly while NGAME made irrelevant predictions. Analysis revealed that the labels predicted by DEXA were all tail labels with 1–5 training points with the average number of training documents for a label being 22. This indicates that the augmentation of text-based representations with auxiliary vectors can particularly help predict data-scarse tail labels more accurately. Similarly, DEXA could predict relevant labels such as 'Acta Senatus', and 'Centuria' whereas most of NGAME's top predictions focused on the repeated token 'Ceasar' (see Table 10). It is notable that it can be challenging to make DEXA's predictions on the basis of label text alone, demonstrating the utility of auxiliary parameters.

7 CONCLUSIONS AND FUTURE DIRECTIONS

This paper presented an alternate architecture to train encoders for large-scale XC applications. The augmentations are light-weight, modular and effective at addressing the semantic gap in XC applications by allowing label representations to not be constrained by label-text alone. Several interesting future directions exist such as better clustering of labels and collaborative learning among label clusters. Setting the number of auxiliary vectors properly, keeping statistical, computational and memory requirements in mind, is key to extracting the benefits of DEXA. As Figure 2 shows, performance does eventually plateau as the number of auxiliary vectors is increased. DEXA also seems to avoid a commonly-observed trade-off between performance on head labels (as indicated by P@k) and performance on tail labels (as indicated by PSP@k). Existing modular XC methods [13, 14] often observe better P@k when using encoder + classifier but better PSP@k when using the encoder alone. However, as Table 15 in the supplementary material indicates, DEXA seems to avoid this trade-off by offering superior P@k as well as comparable or better PSP@k. This merits further investigation.

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

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

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

[11] N. Craswell, D. Campos, B. Mitra, E. Yilmaz, and B. Billerbeck. 2020. ORCAS: 18 Million Clicked Query-Document Pairs for Analyzing Search. arXiv:2006.05324 [cs.IR]

[12] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. arXiv:2003.07820

[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, N. Gupta, D. Saini, A. Soni, Y. Wang, K. Dave, J. Jiao, K. Gururaj, P. Dey, A. Singh, D. Hada, V. Jain, B. Paliwal, A. Mittal, S. Mehta, R. Ramjee, S. Agarwal, P. Kar, and M. Varma. 2023. NGAME: Negative mining-aware mini-batching for extreme classification. In *WSDM*.

[15] 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*.

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

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

[18] 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*.

[19] N. Gupta, P. H. Chen, H.-F. Yu, Cho-J. Hsieh, and I. S. Dhillon. 2022. ELIAS: End-to-End Learning to Index and Search in Large Output Spaces. In *NeurIPS*.

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

[21] 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*.

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

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

[24] 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*.

[25] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis. 2020. Decoupling representation and classifier for long-tailed recognition. In *ICLR*.

[26] 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*.

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

[28] S. Kharbanda, A. Banerjee, E. Schultheis, and R. Babbar. 2022. CascadeXML: Rethinking Transformers for End-to-end Multi-resolution Training in Extreme Multi-label Classification. In *NeurIPS*.

[29] O. Khattab and M. Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*.

[30] Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tieyan Liu, and Arnold Overwijk. 2021. Less is More: Pre-train a Strong Text Encoder for Dense Retrieval Using a Weak Decoder. arXiv:2102.09206 [cs.LG].

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

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

[33] 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*.

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

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

[36] 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*.

[37] 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*.

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

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

[40] 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*.

[41] 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*.

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

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

[44] 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*.

[45] 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.

[46] 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*.

[47] 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*.

[48] T. Renter, A. Borisov, and M. De Rijke. 2016. Siamese CBOW: Optimizing word embeddings for sentence representations. In *ACL*.

[49] 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*.

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

[51] E. Schultheis, M. Wydmuch, R. Babbar, and K. Dembczynski. 2022. On Missing Labels, Long-Tails and Propensities in Extreme Multi-Label Classification. In *KDD*.

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

[53] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957 [cs.CL]

[54] T. Wei, Z. Mao, J.-X. Shi, Y.-F. Li, and M.-L. Zhang. 2022. A Survey on Extreme Multi-label Learning. *arXiv preprint arXiv:2210.03968* (2022).

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

[56] 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*.

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

[58] J. Yang, Z. Liu, S. Xiao, C. Li, D. Lian, S. Agrawal, A. Singh, G. Sun, and X. Xie. 2021. GraphFormers: GNN-nested Transformers for Representation Learning on Textual Graph. In *NeurIPS*.

[59] 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*.

[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*.