UNIVERSITY OF CALIFORNIA
Los Angeles

Compute Efficient Extreme Multi-Label Classification

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Siddhant Kharbanda

2024

ABSTRACT OF THE THESIS

Compute Efficient Extreme Multi-Label Classification

by

Siddhant Kharbanda
Master of Science in Computer Science
University of California, Los Angeles, 2024
Professor Cho-Jui Hsieh, Chair

This thesis focuses on developing novel techniques to improve the performance and computational efficiency of Extreme Multi-label Text Classification (XMC). XMC involves learning a classifier to assign the most relevant subset of labels to an input from a vast label space, often in the order of millions. Over the past decade, XMC has found relevance in numerous large-scale applications, such as query-to-ad-phrase matching in search ads, title-based product recommendation, and prediction of related searches.

There are two major concerns in XMC, catering to the two sides of the problem: one that addresses the short-text query format of the input and the other that addresses the long-text document format of the input. Since its inception, the field has witnessed consistent gains in performance with the scaling of compute in the document format space, while the query format space, due to its online nature, has been limited to computationally efficient smaller models. This thesis addresses both the short-text and long-text nature of XMC. More specifically, the thesis not only provides computationally efficient solutions to the symmetric problem setting, where both input instances and label features are short-text in nature, but also challenges the state-of-the-art in the asymmetric problem, where input instances are long-text, using only a single GPU!

The first chapter proposes Gandalf, a novel approach that leverages a label co-occurrence graph to use label features as additional data points to supplement the training distribution. By exploiting the characteristics of short-text XMC, Gandalf constructs

valid training instances from the label features and uses the label graph to generate corresponding soft-label targets, effectively capturing label-label correlations. Models trained on these new instances, despite being less than half of the original dataset, can outperform models trained on the original dataset, particularly on the PSP@k metric for tail labels. Gandalf can be applied in a plug-and-play manner to various state-of-the-art algorithms, leading to an average 5% relative improvement across 4 benchmark datasets with up to 1.3M labels.

The second chapter addresses the computational cost of conventional XMC models, which often require up to 640GB VRAM to train on the largest public dataset. This high cost is a consequence of calculating the loss over the entire label space. The chapter proposes UniDEC, a loss-independent, end-to-end trainable framework that trains the dual encoder (DE) and classifier together in a unified manner with a multi-class loss, reducing the computational cost by 4-16×. UniDEC employs a pick-some-label (PSL) reduction to compute the loss on only a subset of carefully chosen positive and negative labels in-batch, maximizing their supervisory signals. The proposed framework achieves state-of-the-art results on datasets with millions of labels while being computationally efficient, requiring only a single GPU. UniDEC's state-of-the-art performance and computational efficiency have led to its successful deployment in Microsoft Bing's query-to-ads business, serving over 500 million advertisments in over 100 countries across the EMEA, APAC, and Latin America regions from September 2023 to March 2024.

In summary, this thesis introduces novel techniques to improve the performance and computational efficiency of XMC. The proposed methods, Gandalf and UniDEC, leverage label features, capture label-label correlations, and significantly reduce computational costs while maintaining or improving performance, thus advancing the state-of-the-art in compute efficient extreme multi-label classification.

The thesis of Siddhant Kharbanda is approved.

Nanyun Peng

Aditya Grover

Cho-Jui Hsieh, Committee Chair

University of California, Los Angeles

2024

To my parents, for always believing in me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my thesis advisor, Cho-Jui Hsieh, for his invaluable guidance, unwavering support, and constant encouragement throughout my master's program. His expertise, insights, and constructive feedback have been instrumental in shaping this thesis and helping me grow as a researcher.

I would also like to express my sincere gratitude to my previous research advisors, Rohit Babbar and Donglai Wei, for their incredible support in my research journey. Additionally, I extend my thanks to my collaborators, Erik Schultheis, Devaansh Gupta, Gururaj K, and Pankaj Malhotra, for their time and focused brainstorming sessions that helped refine some ideas.

I am grateful to my thesis committee members, Aditya Grover and Nanyun Peng, for their valuable suggestions and thought-provoking interactions throughout my master's program, which have greatly enhanced the quality of this work and made me a better thinker.

I would like to acknowledge the faculty and staff of the Department of Computer Science at the University of California, Los Angeles, for providing a stimulating academic environment and the resources necessary to complete this research.

My heartfelt thanks go to my fellow graduate students and colleagues, Rohan Wadhwan and Nilay Naharas, for their friendship, stimulating discussions, and collaborative spirit, which made my master's journey more enjoyable and enriching.

I am forever indebted to my parents for their unconditional love, unwavering support, and the countless sacrifices they have made to ensure my success. Their belief in me has been a constant source of motivation.

Lastly, I would like to thank all those who have directly or indirectly contributed to the completion of this thesis. Your support and encouragement have been invaluable.

VITA

2017–2021 B.E. in Computer Science, Birla Institute of Technology And Science, Pilani

PUBLICATIONS

First Author, Gandalf: Learning Label-Label Correlations in Extreme Multi-label Classification via Label Features. *SIGKDD*, 2024

First Author, InceptionXML: A Lightweight Framework with Synchronized Negative Sampling for Short Text Extreme Classification. *SIGIR*, 2023

First Author, CascadeXML: Rethinking Transformers for End-to-end Multi-resolution Training in Extreme Multi-label Classification. *NeurIPS*, 2022

First Author, UniDEC : Unified Dual Encoder and Classifier Training for Extreme Multi-Label Classification. *Under Review*

CHAPTER 1

Learning Label-Label Correlations in Extreme Multi-label Classification via Label Features

1.1 Introduction

Extreme Multilabel Classification (XMC) has found numerous applications in the domains of related searches (Jain et al., 2019), dynamic search advertising (Prabhu et al., 2018b) and recommendation tasks, which require predicting the most relevant results that frequently co-occur together (Chiang et al., 2019; Hu et al., 2020), or are highly correlated to the given product or search query. These tasks are often modeled through embedding-based retrieval-cum-ranking pipelines over millions of possible web page titles, products titles, or ad-phrase keywords forming the label space.

Going beyond conventional tagging tasks for long textual documents consisting of hundreds of words, such as articles in encyclopedia (Partalas et al., 2015), and bio-medicine (Tsatsaronis et al., 2015), contemporary research focus has also widened to settings in which the input is just a short phrase, such as a search query or product title. Propelled by the surge in online search, recommendation, and advertising, applications of short-text XMC ranging from query-to-ad-phrase prediction (Dahiya et al., 2021b) to title-based product-to-product (Mittal et al., 2021a) recommendation have become increasingly prominent.

A major challenge across XMC problems is the extreme imbalance observed in their data distribution. Specifically, these datasets adhere to Zipf's law (Adamic and Huberman, 2002; Ye et al., 2020), i.e., following a long-tailed distribution, where most labels are tail

Datasets	N	L	APpL	ALpP	AWpP
LF-AmazonTitles-131K	294,805	131,073	5.15	2.29	6.92
LF-WikiSeeAlsoTitles-320K	693,082	312,330	4.67	2.11	3.01
LF-WikiTitles-500K	1,813,391	501,070	17.15	4.74	3.10
LF-AmazonTitles-1.3M	2,248,619	1,305,265	38.24	22.20	8.74

Table 1.1: Details of short-text benchmarks with label features. APpL is the avg. points per label, ALpP being avg. labels per point and AWpP is the length i.e. avg. words per point.

labels with very few (≤ 5) positive data-points in a training set spanning $\geq 10^6$ total data points (Table 1.1). With so few positive examples, training a successful classifier on these labels purely from instance-to-label pairs seems an insurmountable challenge. Therefore, recent methods have begun to incorporate additional data sources.

Label features and label co-occurrence In many of the settings listed above, labels are not just featureless integers, but do have a semantic meaning in and of themselves. For example, when matching products, each product ID could be associated with the name of the product. This is particularly attractive in the short-text setting, when both inputs and labels come from the *same* space of short phrases. Consequently, while earlier work mostly focused on the nuances of short-text inputs (Dahiya et al., 2021b; Kharbanda et al., 2023), more recent methods have successfully incorporated the short-text label descriptors into their pipeline (Mittal et al., 2021a,b; Dahiya et al., 2021a, 2023a).

Yet, this still seems to underutilize the wealth of information present in label features. In particular, we demonstrate that it is possible to train a classifier using *only* label information, that is, without ever presenting to it any of the training instances, and *outperform* the same classifier trained on the original training data on tail labels. This surprising feat is enabled by the exploitation of label co-occurrence information.

In particular, using the interchangability of label features and instances, instead of aiming for contrastive learning (Dahiya et al., 2021a), we want to use the label features as additional, supervised training points. However, this requires them to be associated with some apriori unknown label vector. In order to generate training targets, we make

the assumption that the probability of a label j being relevant for the textual feature of another label i, is equal to the conditional probability of observing j, given that i is also a relevant label.

Contributions This insight yields a simple method, *Gandalf* (**G**raph **A**ugme**N**ted **DA**ta with **L**abel **F**eatures), which exploits the unique setting of short-text XMC in a novel manner to generate additional training data in order to alleviate the data scarcity problem. As a data-centric approach, it is independent of the specific model architecture, enabling its application to a wide range of both current and potential future state-of-the-art models. The unchanged model architecture also implies that not only the model inference latency remains unchanged, but also peak memory consumption required during training is unaffected, contrary to some model-based approaches that incorporate label metadata (Mittal et al., 2021a; Dahiya et al., 2021a; Chien et al., 2023).

The additional training instances lead to overall longer training time. Nonetheless, when keeping the compute budget fixed, we can observe *Gandalf* significantly outperforming the original dataset. When trained until convergence, we show an average of 5% improvement on 5 state-of-the-art extreme classifiers across 4 public short-text benchmarks, with some settings seeing gains up to 30%. In this way, XMC methods which inherently do not leverage label features can beat or perform on par with strong baselines which either employ elaborate training pipelines (Dahiya et al., 2021a), large transformer encoders (You et al., 2019; Zhang et al., 2021c; Dahiya et al., 2023a) or make heavy architectural modifications (Mittal et al., 2021a,b) to leverage label features.

Finally, we show that Gandalf could be considered an extension of the GLaS (Guo et al., 2019) regularizer to the label feature setting. We interpret it as tuning the bias-variance trade-off, where the additional error introduced by inaccurate additional training data is more then compensated for by the decrease is variance, especially for extremely noise tail labels (Buvanesh et al., 2024).

1.2 Preliminaries

For training, we have available a multi-label dataset $\mathcal{D} = \left(\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N, \{\mathbf{z}_l\}_{l=1}^L\right)$ comprising of N data points. Each $i \in [N]$ is associated with a small ground truth label vector $\mathbf{y}_i \in \{0,1\}^L$ from $L \sim 10^6$ possible labels. Further, $\mathbf{x}_i, \mathbf{z}_l \in \mathcal{X}$ denote the textual descriptions of the data point i and the label l which, in this setting, derive from the same vocabulary universe $\mathcal{V}$ (Dahiya et al., 2021a). The goal is to learn a parameterized function $f : \mathbf{x}_i \mapsto \mathbf{y}_i$.

One-vs-All Classification (OvA) A common strategy for handling this learning problem is to map instances and labels into a common Euclidean space $\mathcal{E} = \mathbb{R}^d$, in which the relevance $s_l(\mathbf{x})$ of a label l to an instance is scored using an inner product, $s_l(\mathbf{x}) = \langle \Phi(\mathbf{x}), \mathbf{w}_l \rangle$. Here, $\Phi(\mathbf{x})$ is the embedding of instance $\mathbf{x}$, and $\mathbf{w}_l$ the l'th column of the weight matrix $\mathbf{W}$.

The prediction function selects the k highest-scoring labels, $f(\mathbf{x}) = \text{top}_k\left(\langle \Phi(\mathbf{x}), \mathbf{W} \rangle\right)$. Training is usually handled using the *one-vs-all* paradigm, which applies a binary loss function ℓ to each entry in the score vector. In practice, performing the sum over all labels for each instance is prohibitively expensive, so the sum is approximated by a shortlist of labels $\mathcal{S}(\mathbf{x}_i)$ that typically contains all the positive labels, and only those negative labels which are expected to be challenging for classification (Dahiya et al., 2021a,b, 2023a; Zhang et al., 2021c; Kharbanda et al., 2023):

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}[\Phi, \mathbf{W}] &= \sum_{i=1}^{N} \sum_{l=1}^{L} \ell(\mathbf{y}_{il}, \langle \Phi(\mathbf{x}), \mathbf{w}_l \rangle) \\ &\approx \sum_{i=1}^{N} \sum_{l \in \mathcal{S}(\mathbf{x}_i)} \ell(\mathbf{y}_{il}, \langle \Phi(\mathbf{x}), \mathbf{w}_l \rangle) \,. \end{aligned} \tag{1.1}$$

Even though these approaches have been used with success, they still struggle in learning good embeddings $\mathbf{w}_l$ for tail labels: A classifier that learns solely based on instance-label pairs has little chance of learning similar label representations for labels

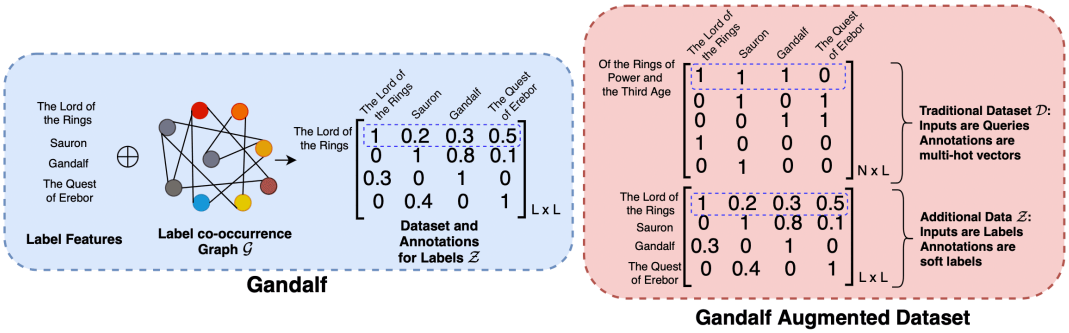


Figure 1.1: Gandalf augments the training dataset $\mathcal{D}$ by generating soft targets for each label based on label co-occurrence statistics. These additional datapoints $\mathcal{Z}$ are simply concatenated to the traditional dataset for training.

that do not co-occur within the dataset, even though they might be semantically related. Consequently, training can easily lead to overfitting even with simple classifiers (Guo et al., 2019).

Label Features To reduce the generalization gap, regularization needs to be applied to the label weights $\mathbf{W}$, either explicitly as a new term in the loss function (Guo et al., 2019), or implicitly through the inductive biases of the network structure (Mittal et al., 2021a,b) or by a learning algorithm (Dahiya et al., 2021a, 2023a). These approaches incorporate additional label metadata – *label features* – to generate the inductive biases. For short-text XMC, these features themselves are often short textual description, coming from the same space as the instances, as the following examples, taken from (i) LF-AmazonTitles-131K (recommend related products given a product name) and (ii) LF-WikiTitles-500K (predict relevant categories, given the title of a Wikipedia page) illustrate:

Example 1: For *"Mario Kart: Double Dash!!"* on Amazon, we have available: *Mario Party 7* | *Super Smash Bros Melee* | *Super Mario Sunshine* | *Super Mario Strikers* as the recommended products.

Example 2: For the *"2022 French presidential election"* Wikipedia page, we have the available categories: *April 2022 events in France* | *2022 French presidential election* |

2022 elections in France | Presidential elections in France. Further, a google search of the same query, leads to the following related searches - *French election 2022 - The Economist | French presidential election coverage on FRANCE 24 | Presidential Election 2022: A Euroclash Between a "Liberal... | French polls, trends and election news for France*, amongst others.

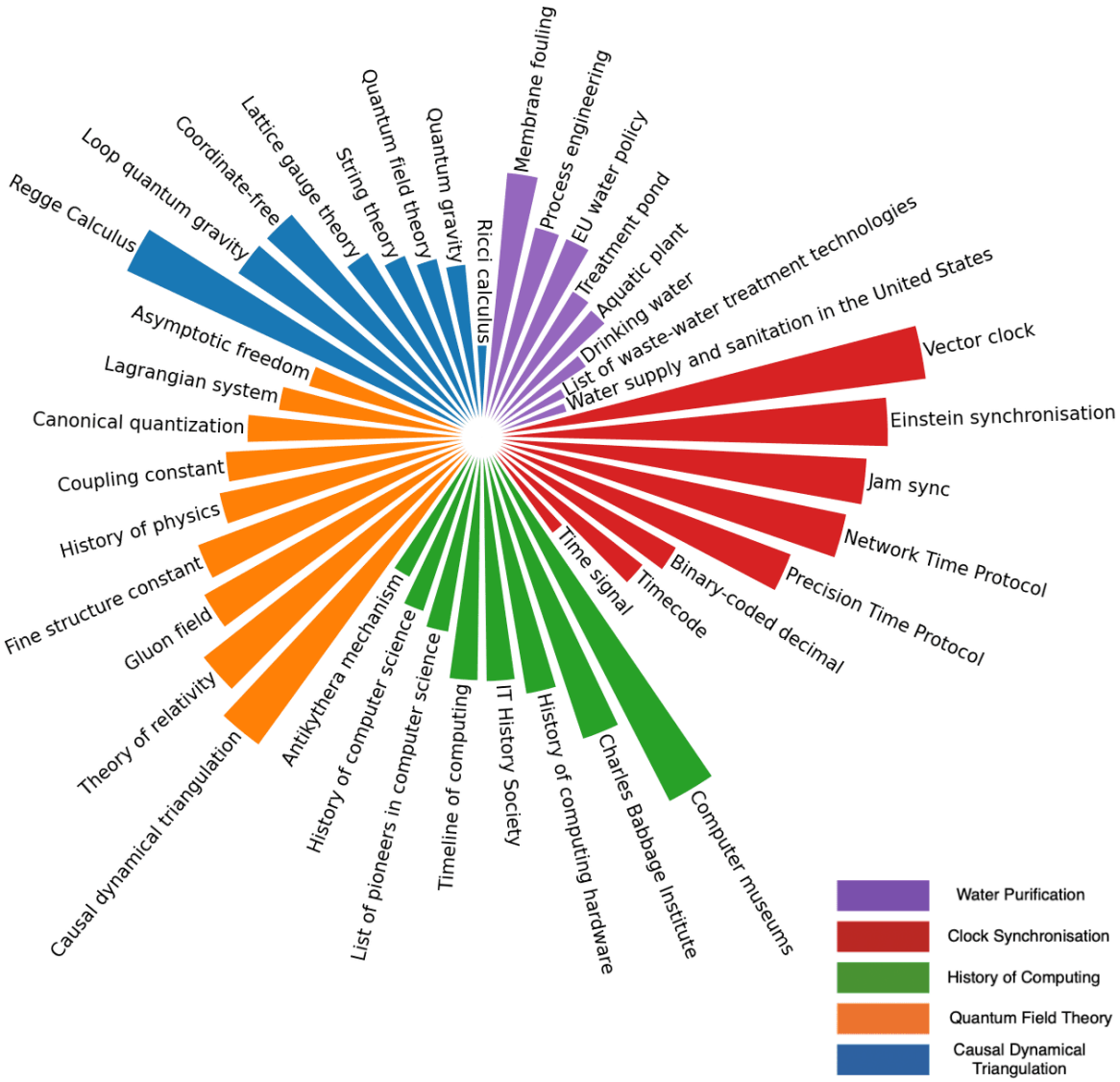


Figure 1.2: Correlations between labels and their first-order neighbours, as found by the label co-occurrence on the LF-WikiTitles-500K dataset. The legend shows the selected label, the bar chart shows the degree of correlation with its neighbouring labels. Correlated labels often share tokens with each other and/or may be used in the same context.

In view of these examples, one can affirm two important observations: (i) the short-text XMC problem indeed requires recommending similar items which are either highly correlated or co-occur frequently with the queried item, and (ii) the queried item and the corresponding label-features form an "equivalence class" and convey similar intent (Dahiya et al., 2021a). For example, a valid news headline search should either result in a page mentioning the same headline or similar headlines from other media outlets (see Example 2). As a result, it can be argued that data instances are *interchangeable* with their respective labels' features. Exploiting this interchangeability of label and instance text, Dahiya et al. (2021a, 2023a) proposes to tie encoder and decoder together and require $\mathbf{w}_l = \Phi(\mathbf{z}_l)$. While indeed yielding improved test performance, the condition $\mathbf{w}_l = \Phi(\mathbf{z}_l)$ turns out to be too strong, and it has to allow for some fine-tuning corrections $\boldsymbol{\eta}_l$, yielding $\mathbf{w}_l = \Phi(\mathbf{z}_l) + \boldsymbol{\eta}_l$. Consequently, training of SiameseXML and NGAME is done in two stages: a contrastive loss is minimized, followed by fine-tuning with a classification objective.

Label correlations Label-label dependencies can appear in multi-label classification in two different forms: *Conditional* label correlations, and *marginal* label correlations (Dembczyński et al., 2012). In the conditional case, label dependencies are considered conditioned on each individual query, that is, they are independent if[1]

$$\mathbb{P}[\mathbf{Y} \mid X] = \prod_j \mathbb{P}[Y_j \mid X] \tag{1.2}$$

As an example, consider the search query *"Jaguar"*: If we know just this search term, the results pertaining to both, the car brand and the animal, are likely to be relevant. However, knowing that during a particular instance of this search, the user was interested in the animal, one can conclude that car-based labels are less likely to be relevant. In this way, the presence of one label gives information *beyond* what can be extracted just from the search query.

[1]Capital X and $\mathbf{Y}$ denote the random variables associated with instance and labels, rsp.

On the other hand, similar labels will generally appear together. Taking example 2 from the previous section, labels *"2022 events in France"* and *"2022 elections in France"* will have an above-random chance of occurring together; however, that information is already carried in the query *"2022 French presidential election"*, so the presence of one of these labels doesn't provide any new information, *given* the query. In that sense, labels are marginally independent if

$$\mathbb{P}[\mathbf{Y}] = \prod_j \mathbb{P}[Y_j] \ . \tag{1.3}$$

Given an instance, OvA classifiers generate scores independently for all labels. Thus, they are *fundamentally incapable* of modelling conditional label dependence. However, as standard performance metrics (P@k, PSP@k) are also decomposable into independent contributions of each label, that is, they can be expressed purely in terms of label marginals, they are similarly incapable of detecting whether a classifier models conditional label dependence (Dembczyński et al., 2012).

This means that, as long as we want to focus on these standard metrics (and not on inter-dependency aware losses such as Hüllermeier et al. (2022)), we only need to care about marginal correlations. At first glance, this seems trivial: It can be shown that an OvA classifier, trained using a proper loss, is consistent for P@k (Menon et al., 2019). Unfortunately, consistency only tells us that, in the limit of infinite training data, we will get a Bayes-optimal classifier. However, in practice, the XMC setting is very far from infinite data—most tail labels will have less than five positive training examples.

Thus, the question we aim to tackle here is: *Can we exploit knowledge about marginal label correlations to improve training in the data-scarce regime of long-tailed multi-label problems?*

1.3 Gandalf: Learning From Label-Label Correlations

By combining marginal label correlations with label features, we can extend the *self-annotation postulate* of Dahiya et al. (2021a) to:

Postulate 1.3.1. *Label-feature Annotation: Given a label j with label-features $\mathbf{z}_j$, we posit that if these features are posed as a data point, its labels should follow the marginal label correlations, that is*

$$\mathbb{P}[Y_i = 1 \mid X = \mathbf{z}_j] \approx \mathbb{P}[Y_i = 1 \mid Y_j = 1] \ . \tag{1.4}$$

Note that this reduces to self-annotation by setting $i = j$, in which case (1.4) becomes $\mathbb{P}[Y_j \mid \mathbf{z}_j] \approx \mathbb{P}[Y_j \mid Y_j] = 1$.

In words, this means that, if the presence of label j indicates that label i would occur with a certain probability for that same instance, then we assume that this probability is also how likely that label i is to be relevant to a data point that consists of the label features of label j. The right side of (1.4) can be written as

$$\mathbb{P}[Y_i = 1 \mid Y_j = 1] = \mathbb{P}[Y_i = 1, Y_j = 1] \,/\, \mathbb{P}[Y_j = 1] \ . \tag{1.5}$$

Thus, we can use the co-occurrence statistics $G_{ij} \coloneqq \mathbb{P}[Y_i = 1, Y_j = 1]$ to calculate the conditionals, and thus apply a plug-in approach using empirical co-occurrence:

$$\mathbb{P}[Y_i = 1 \mid Y_j = 1] \approx \frac{\hat{G}_{ij}}{\hat{G}_{jj}} \,, \text{where}, \hat{G}_{ij} \coloneqq \sum_{s=1}^{n} y_{si} y_{sj} \ .$$

Of course, in the data-scarce XMC regime, the co-occurrence matrix $\boldsymbol{G}$ will be very noisy. In practice, we empirically find it beneficial to threshold the soft labels at δ, so that label features as data-points are annotated by:

$$y_{ij}^{\mathrm{G}} \coloneqq \begin{cases} \hat{G}_{ij}/\hat{G}_{jj} \text{ if } & \hat{G}_{ij}/\hat{G}_{jj} > \delta \\ 0 & \text{otherwise} \end{cases} \ . \tag{1.6}$$

By approximating the left-hand side of (1.4) using a parameterized model Ψ, and taking the empirical co-occurrence as a noise estimate for the right-hand side, we can turn this equation into a (surrogate) machine-learning task. This is the same problem as the original XMC task (1.1), applied to a different dataset $\mathcal{Z} = \{(\mathbf{z}_i, \mathbf{y}_i^{\mathrm{G}})\}_{i=1}^{L}$. That is, we want to optimize

$$\mathcal{L}_{\mathcal{Z}}[\Psi, \mathbf{W}] \coloneqq \sum_{i,j=1}^{L} y_{ij}^{\mathrm{G}}, \langle \Psi(\mathbf{z}_j), \mathbf{w}_i \rangle) \,. \tag{1.7}$$

In Table 1.2, we present results for training on this surrogate task (row "Training on $\mathcal{Z}$"), when evaluating the resulting classifier on the original test set. The results are striking, and provide a strong confirmation of the equivalence principle between label features and input texts: Even though this model has *never* seen any actual training instance, it performs adequate (AmazonTitles) or better (WikiSeeAlsoTitles) than the original model in terms of precision at k. Looking at PSP, which gives more weight to tail labels, it actually outperforms the original model, in some cases with a large margin.

This tells us that we can, in fact, identify the two encoders in equations 1.1 and 1.7, $\Psi \equiv \Phi$, and train a single model on the combined dataset $\mathcal{G} = \mathcal{D} \cup \mathcal{Z}$, as illustrated in Figure 1.1. This combination of data yields strong improvements on both regular and tail-label performance metrics.

	LF-AmazonTitles-131K				**LF-WikiSeeAlsoTitles-320K**			
Training Data	**P@1**	**P@5**	**PSP@1**	**PSP@5**	**P@1**	**P@5**	**PSP@1**	**PSP@5**
original $\mathcal{D}$	35.62	17.35	27.53	37.50	21.53	10.66	13.06	16.33
surrogate $\mathcal{Z}$	29.68	16.04	28.76	38.27	22.88	12.44	22.03	25.55
$\mathcal{G} = \mathcal{Z} \cup \mathcal{D}$	**43.52**	**20.92**	**36.96**	**47.64**	**31.31**	**16.22**	**24.31**	**28.83**
$\mathcal{U}(\mathcal{G}, N)$	38.46	18.52	32.29	41.59	25.93	13.34	19.75	23.57
$\mathcal{Z}^1 \cup \mathcal{D}$	37.59	18.18	30.75	40.06	24.43	12.15	16.89	20.02
$\mathcal{G}$, $y_{ll'} = \sigma(S_{ll'})$	42.80	20.49	37.01	46.73	30.28	15.43	23.97	28.45

Table 1.2: Experiments showing the quality of the datasets created with label features on InceptionXML. While the baseline is surpassed by training on the combined dataset $\mathcal{G}$, it is also beaten by training on $\mathcal{Z}$, where $|\mathcal{Z}| < |\mathcal{N}|/2$, underscoring its quality.

1.3.1 Bias-Variance Trade-off

This improvement cannot be explained by the increased training set size $|\mathcal{G}| = N + L$ alone, as we can show with the following simple experiment: We generate a new dataset $\mathcal{G}' \sim \mathcal{U}(\mathcal{G}, N)$ by uniformly sampling (without replacement) from the combined dataset a subset that has the same size as the original training set $|\mathcal{D}| = N$. Table 1.2 shows that this already leads to significant improvements over the original training set.

To explain this phenomenon, we note that this augmented data is qualitatively slightly different from the original training instances: the empirical co-occurrence matrix $\hat{\boldsymbol{G}}$ provides *soft* labels $\mathbf{y}_i^{\mathrm{G}}$ as training targets. XMC dataset exhibit high variance (Babbar and Schölkopf, 2019; Buvanesh et al., 2024) because of the long tail labels, whereas the soft labels of the augmented points provide a much smoother training signal. On the other hand, they are based on the approximation of Postulate 1.3.1, and as such, will introduce some additional bias into the method, essentially leading to a highly favourable bias-variance trade-off.

In fact, the reduction in variance is so helpful to the training process that even switching out (1.4) with one-hot labels based purely on the self-annotation principle ($y_{ij}^{\mathrm{SA}} \coloneqq \mathbb{1}[i = j]$ such that $\mathcal{Z}^1 = \{(\mathbf{z}_i, \mathbf{y}_i^{\mathrm{SA}})\}_{i=1}^{L}$), thus considerably increasing the bias in the generated data, we still get significant improvements over just using the original training data (Table 1.2).

1.3.2 Connection to GLaS regularization

In order to derive a model for $\mathbb{P}[Y_{l'} = 1 \mid X = \mathbf{z}_l]$, we can take inspiration from the GLaS regularizer (Guo et al., 2019). This regularizer tries to make the Gram matrix of the label embeddings $\langle \mathbf{w}_i, \mathbf{w}_j \rangle$ reproduce the co-occurrence statistics of the labels $\mathbf{S}$,

$$\mathcal{R}_{\mathrm{GLaS}}[\mathbf{W}] = L^{-2} \sum_{i=1}^{L} \sum_{j=1}^{L} \left(\langle \mathbf{w}_i, \mathbf{w}_j \rangle - S_{ij} \right)^2 . \tag{1.8}$$

Here, $\mathbf{S}$ denotes the symmetrized conditional probabilities,

$$\begin{aligned} S_{ij} &\coloneqq 0.5(\mathbb{P}[Y_i = 1 \mid Y_j = 1] + \mathbb{P}[Y_j = 1 \mid Y_i = 1]) \\ &\approx 0.5(\hat{G}_{ij}/\hat{G}_{jj} + \hat{G}_{ij}/\hat{G}_{ii})\,. \end{aligned} \tag{1.9}$$

By the self-proximity postulate (Dahiya et al., 2021a), we can assume $\mathbf{w}_l \approx \Phi(\mathbf{z}_l)$. For a given label feature instance with target soft-label $(\mathbf{z}_l, y_{ll'}^{\text{GLaS}})$, the training will try to minimize $\ell(\langle \Phi(\mathbf{z}_l), \mathbf{w}_{l'} \rangle, y_{ll'}^{\text{GLaS}})$. To be consistent with Equation 1.8, we therefore want to choose $y_{ll'}^{\text{GLaS}}$ such that $S_{ll'} = \arg\min \ell(\cdot, y_{ll'}^{\text{GLaS}})$. This is fulfilled for $y_{ll'}^{\text{GLaS}} = \sigma(S_{ll'})$ for ℓ being binary cross-entropy, where σ denotes the logistic function.

While the soft targets generated this way slightly differ from the ones of (1.6), as already observed, the bias introduced by mildly incorrect training targets is offset by far by the variance reduction, and we find that this version performs only slightly worse than *Gandalf* (Table 1.2).

1.4 Main Results & Discussion

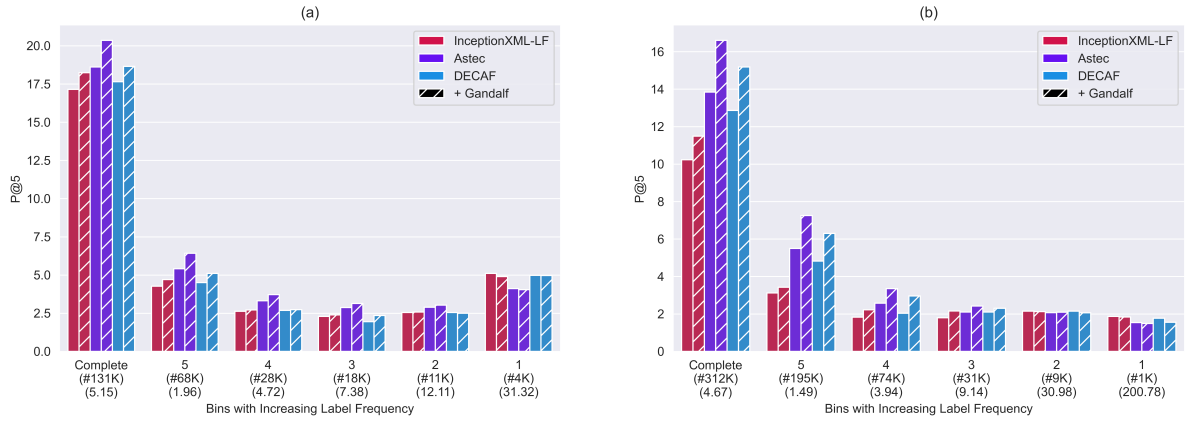
Benchmarks, Baseline and Metrics We benchmark our experiments on 4 standard public datasets, the details of which are mentioned in Table 1.1. To test the generality and effectiveness of our proposed *Gandalf*, we apply the algorithm across a variety of state-of-the-art short-text extreme classifiers. These consist of (i) base frugal models -Astec (Dahiya et al., 2021b) and InceptionXML (Kharbanda et al., 2023) - which do not, by default, leverage label text information, (ii) Decaf (Mittal et al., 2021a), Eclare (Mittal et al., 2021b) and InceptionXML-LF which equip the base models with additional encoders to make use of label text and label correlation information and, (iii) Ngame + Renee - consisting of Renee (Jain et al., 2023), which makes CUDA optimizations to train BCE loss over a classifier for L labels without a shortlist. The transformer encoder is initialized with pre-trained Ngame (M1, dual encoder) (Dahiya et al., 2023a). We measure the performance using standard metrics P@k, its propensity-

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
	LF-AmazonTitles-131K						LF-AmazonTitles-1.3M					
SiameseXML	41.42	**30.19**	21.21	35.80	40.96	46.19	49.02	42.72	38.52	27.12	30.43	32.52
Astec	37.12	25.20	18.24	29.22	34.64	39.49	48.82	42.62	38.44	21.47	25.41	27.86
+ *Gandalf*	43.95	29.66	21.39	37.40	43.03	48.31	53.02	46.13	41.37	27.32	31.20	33.34
Decaf	38.40	25.84	18.65	30.85	36.44	41.42	50.67	44.49	40.35	22.07	26.54	29.30
+ *Gandalf*	42.43	28.96	20.90	35.22	42.12	47.61	53.02	46.65	42.25	25.47	30.14	32.83
Eclare	40.46	27.54	19.63	33.18	39.55	44.10	50.14	44.09	40.00	23.43	27.90	30.56
+ *Gandalf*	42.51	28.89	20.81	35.72	42.19	47.46	53.87	47.45	43.00	28.86	32.90	35.20
InceptionXML	36.79	24.94	17.95	28.50	34.15	38.79	48.21	42.47	38.59	20.72	24.94	27.52
+ *Gandalf*	44.67	30.00	21.50	37.98	43.83	48.93	50.80	44.54	40.25	25.49	29.42	31.59
InceptionXML-LF	40.74	27.24	19.57	34.52	39.40	44.13	49.01	42.97	39.46	24.56	28.37	31.67
+ *Gandalf*	43.84	29.59	21.30	38.22	43.90	49.03	52.91	47.23	42.84	**30.02**	33.18	35.56
Ngame + Renee	**46.05**	**30.81**	**22.04**	38.47	44.87	50.33	56.04	49.91	45.32	28.54	**33.38**	**36.14**
+ *Gandalf*	45.86	30.53	21.79	**40.49**	**45.83**	**50.96**	**56.88**	**50.24**	**45.47**	26.56	31.69	34.60
	LF-WikiSeeAlsoTitles-320K						LF-WikiTitles-500K					
SiameseXML	31.97	21.43	16.24	**26.82**	28.42	30.36	42.08	22.80	16.01	23.53	21.64	21.41
Astec	22.72	15.12	11.43	13.69	15.81	17.50	44.40	24.69	17.49	18.31	18.25	18.56
+ *Gandalf*	31.10	21.54	16.53	23.60	26.48	28.80	45.24	25.45	18.57	21.72	20.99	21.16
Decaf	25.14	16.90	12.86	16.73	18.99	21.01	44.21	24.64	17.36	19.29	19.82	19.96
+ *Gandalf*	31.10	21.60	16.31	24.83	27.18	29.29	45.27	25.09	17.67	22.51	21.63	21.43
Eclare	29.35	19.83	15.05	22.01	24.23	26.27	44.36	24.29	16.91	21.58	20.39	19.84
+ *Gandalf*	31.33	21.40	16.31	24.83	27.18	29.29	45.12	24.45	17.05	24.22	21.41	20.55
InceptionXML	23.10	15.54	11.52	14.15	16.71	17.39	44.61	24.79	19.52	18.65	18.70	18.94
+ *Gandalf*	32.54	22.15	16.86	25.27	27.76	30.03	45.93	25.81	**20.36**	21.89	21.54	22.56
InceptionXML-LF	28.99	19.53	14.79	21.45	23.65	25.65	44.89	25.71	18.23	23.88	22.58	22.50
+ *Gandalf*	33.12	22.70	17.29	26.68	**29.03**	**31.27**	**47.13**	**26.87**	19.03	**24.12**	**23.92**	**23.82**
Ngame + Renee	30.79	20.65	15.57	20.81	24.46	27.05	-	-	-	-	-	-
+ *Gandalf*	**33.92**	**23.11**	**17.58**	24.15	26.23	30.89	-	-	-	-	-	-

Table 1.3: Results showing the effectiveness of *Gandalf* on state-of-the-art extreme classifiers. The best results are in **bold**. Results for Ngame + Renee have been used from their publication, however, have not been reported for LF-WikiTitles-500K.

scored variant, PSP@k (Jain et al., 2016; Qaraei et al., 2021), and coverage@k (Schultheis et al., 2022, 2024).

Improvements on tail labels We perform a quantile analysis across 2 datasets – LF-AmazonTitles-131K and the LF-WikiSeeAlso- Titles-320K (Figure 1.3) with InceptionXML – where we examine performance (contribution to P@5 metric) over 5 equi-voluminous bins based on increasing order of mean label frequency in the training dataset. Consequently, performance on head labels can be captured by the bin #1 and that of tail labels by bin #5. We note that introducing the additional training data with *Gandalf* consistently improves the performance across all label frequencies, with more profound gains on bins with more tail labels. This is further verified by significant



(a) Results on LF-AmazonTitles-131K (b) Results on LF-WikiSeeAlsoTitiles-320K

Figure 1.3: GANDALF demonstrating improvements on the P@5 metric across various methods, separated into tail, torso and head labels. On the x axis, the middle row indicates the number of labels in the bin, and the lowest row denotes the average number of positives per label in that bin. Improvements in earlier bins (5 - 3) denote gains in tail label performance.

performance boosts, with base models showing upto 11% improvements in the PSP@k metrics in Table 1.3.

Gandalf vs Architectural Additions (*LTE*, *GALE*) The first formal attempt to externally imbue the model with label information was made with DECAF, which essentially equips the base model ASTEC with another base encoder (*LTE*) to learn label text ($\mathbf{z}_l$) embeddings along with the classifier. The second attempt, in the form of ECLARE, builds upon DECAF by adding another base encoder (*GALE*) to process and *externally* capture label correlation information. To make our claim more general, we also evaluate on INCEPTIONXML-LF, which consist of the same extensions on a more recent base model INCEPTIONXML (Kharbanda et al., 2023) with *LTE* and *GALE* components (that ECLARE adds over ASTEC). While such architectural modifications help capture higher order query-label relations and increase empirical performance, they also increase both, training time and the peak GPU memory required during training by $\sim 3\times$.

As *Gandalf* is a data-centric approach, the memory overhead is eliminated by default. Further, we find that (i) DECAF and ECLARE still benefit from using *Gandalf* augmented

data implying architectural modifications are complementary to *Gandalf*. However, (ii) simply using *Gandalf* augmented data enables base models Astec and InceptionXML outperform themselves by up to 30% and perform nearly at par with their more architecturally equipped counter parts Eclare and InceptionXML-LF. While we posit that *Gandalf* and *GALE* learn complementary data relations, both our quantitative (Table 1.3) and qualitative (Table 1.5, Figure 1.3) results show that *Gandalf* is more effective and efficient at capturing these relations (specifically, label correlations) compared to the latter.

Beyond model performances We can also extract dataset specific insights with Gandalf from Table 1.3. Significant improvements on top of the base algorithm are particularly observed on LF-AmazonTitles-131K and LF-WikiSeeAlsoTitles-320K. In contrast, improvements on LF-WikiTitles-500K remain relatively mild. We attribute this to the density of the datasets. Specifically, while the former datasets consist of ∼5 training instances per label, the latter consists of ∼17. We posit a higher query-label density enables algorithms to inherently learn sufficient label-label correlations from existing data. However, we further see that using *Gandalf* is effective for LF-AmazonTitles-1.3M, the largest public benchmark for XMC with label features. Here, even though average training instances per label is ∼38, the average number of labels per instance is ∼22, as compared to maximum of ∼4 on other datasets.

Gandalf vs Siamese Learning Consequently, the third attempt made at capturing label correlations via SiameseXML, which essentially replaces the surrogate training task in Astec with a two-tower siamese learning framework. As argued in § 1.2, the condition $\mathbf{w}_l = \Phi(\mathbf{z}_l)$ turns out to be too strong, and consequently training of SiameseXML and NGAME is done in two stages. Initially, a contrastive loss needs to be minimized, followed by fine-tuning with a classification objective which allows for some fine-tuning corrections $\boldsymbol{\eta}_l$, yielding $\mathbf{w}_l = \Phi(\mathbf{z}_l) + \boldsymbol{\eta}_l$. On the other hand, *Gandalf* simply extends training data to learn from a-priori label co-occurrence data in a supervised manner.

Notably (from Table 1.3), Astec + *Gandalf* outperforms SiameseXML by 5-10% on Amazon datasets, while performing at par on Wikipedia datasets.

Applying Gandalf to Two-tower approaches Although we propose Gandalf as a method suitable for training classifiers, it can also be used leveraged alongside two-tower approaches, like Ngame. This is done by first extending the dual encoder with a scalable classifier with Renee, which simply trains OvA classifiers on top of the base model. Using Gandalf augmented data during this extension leads to significant improvements, more prominently on the LF-WikiSeeAlsoTitles-320K and LF-AmazonTitles-131K datasets.

Coverage Results Coverage is an important metric in XMC as it demonstrates the ability of the model to predict tail labels effectively. We provide coverage results on InceptionXML in Table 1.4, demonstrating that Gandalf learns to predict labels which were previously not being predicted at all. This phenomenon can also be seen in the qualitative results Table 1.5.

Method	C@1	C@3	C@5	C@1	C@3	C@5
	LF-AmazonTitles-131K			**LF-WikiSeeAlsoTitles-320K**		
InceptionXML	22.33	39.98	46.29	7.54	15.11	18.93
+ Gandalf	**31.04**	**51.63**	**58.03**	**13.28**	**26.01**	**32.21**

Table 1.4: Coverage Results on InceptionXML with Gandalf.

Qualitative Results We further analyse qualitative examples via the top 5 predictions obtained by training the base encoders with and without *Gandalf* augmented data points in Table 1.5. Notably, we can observe that queries with even a single keyword (*Oat*), which have no correct predictions without Gandalf, result in 100% correct predictions with it. Furthermore, even the quality of incorrect predictions improves and we suspect these labels are more likely to be *missed true positives*. (Jain et al., 2016) For example, in case of "Lunar Orbiter program", the only incorrect *Gandalf* predictions are "Lunar Orbiter 3", "Lunar Orbiter 5" and "Pioneer program" (US lunar and planetary space

Method	Datapoint	Baseline Predictions	*Gandalf* Predictions
InceptionXML-LF	Topological group	Pontryagin duality, Topological order, Topological quantum field theory, Topological quantum number, Quantum topology	Compact group, Haar measure, Lie group, Algebraic group, Topological ring
Decaf		Topological quantum computer, Topological order, Topological quantum field theory, Topological quantum number, Quantum topology	Compact group, Haar measure, Lie group, Algebraic group, Topological ring
Eclare		Topological quantum computer, Topological order, Topological quantum field theory, Topological quantum number, Quantum topology	Compact group, Topological order, Lie group, Algebraic group, Topological ring
InceptionXML-LF	Oat	List of lighthouses in Scotland, List of Northern Lighthouse Board lighthouses, Oatcake, Communes of the Finistere department, Oat milk	Oatcake, Oatmeal, Oat milk, Porridge, Rolled oats
Decaf		Oatcake, Oatmeal, Design for All (in ICT), Oatley Point Reserve, Oatley Pleasure Grounds	Oatcake, Oatmeal, Oat milk, Porridge, Rolled oats
Eclare		Oatmeal, Oat milk, Parks in Sydney, Oatley Point Reserve, Oatley Pleasure Grounds	Oatcake, Porridge, Rolled oats, Oatley Point Reserve, Oatley Pleasure Grounds
InceptionXML-LF	Lunar Orbiter program	Lunar Orbiter Image Recovery Project, Lunar Orbiter 3, Lunar Orbiter 5, Chinese Lunar Exploration Program, List of future lunar missions	Surveyor program, Luna programme, Lunar Orbiter Image Recovery Project, Lunar Orbiter 3, Lunar Orbiter 5
Decaf		Exploration of the Moon, List of man-made objects on the Moon, Lunar Orbiter Image Recovery Project, Lunar Orbiter 3, Lunar Orbiter 5	Exploration of the Moon, Apollo program, Surveyor program, Luna programme, Lunar Orbiter program
Eclare		Exploration of the Moon, Lunar Orbiter program, Lunar Orbiter Image Recovery Project, Lunar Orbiter 3, Lunar Orbiter 5	Exploration of the Moon, Pioneer program, Surveyor program, Luna programme, Lunar Orbiter program
InceptionXML-LF	Grand Lake, Colorado	Colorado metropolitan areas, Front Range Urban Corridor, Outline of Colorado, Index of Colorado-related articles, State of Colorado	Colorado metropolitan areas, Outline of Colorado, Index of Colorado-related articles, Colorado cities and towns, Colorado counties
Decaf		Colorado metropolitan areas, Front Range Urban Corridor, State of Colorado, Colorado municipalities, National Register of Historic Places listings in Grand County, Colorado	Outline of Colorado, State of Colorado, Colorado cities and towns, Colorado municipalities, Colorado counties
Eclare		State of Colorado, Colorado cities and towns, Colorado counties, National Register of Historic Places listings in Grand County, Colorado, Grand County, Colorado	Outline of Colorado, Index of Colorado-related articles, State of Colorado, Colorado cities and towns, Colorado counties
InceptionXML-LF	Armed Forces of Saudi Arabia	Royal Saudi Air Defense, Royal Saudi Strategic Missile Force, Saudi Royal Guard Regiment, Terrorism in Saudi Arabia, Capital punishment in Saudi Arabia	Military of Saudi Arabia, Royal Saudi Air Force, Royal Saudi Air Defense, Royal Saudi Strategic Missile Force, King Khalid Military City
Decaf		Saudi Arabian-led intervention in Yemen, Saudi-led intervention in Bahrain, Human rights in Saudi Arabia, Legal system of Saudi Arabia, Joint Chiefs of Staff (Saudi Arabia)	Royal Saudi Air Force, Royal Saudi Navy, Royal Saudi Air Defense, Royal Saudi Strategic Missile Force, Saudi Arabian National Guard
Eclare		List of armed groups in the Syrian Civil War, Military of Saudi Arabia, Royal Saudi Strategic Missile Force, King Khalid Military City, Joint Chiefs of Staff (Saudi Arabia)	Military of Saudi Arabia, Royal Saudi Air Defense, Royal Saudi Strategic Missile Force, King Khalid Military City, Saudi Royal Guard Regiment

Table 1.5: Qualitative predictions from the LF-WikiSeeAlsoTitles-320K dataset. Labels indicate mispredictions.

programs). Additionally, we show semantic similarity between the annotated labels with $\mathcal{G}$, and the original label in Figure 1.2.

1.5 Ablations & Computational Analysis

Gandalf, is a data-centric approach that does not increase the computational cost during inference. While the inclusion of label features - which can often run in the order of millions - as additional data points might seem to increase the computational cost during training, through a series of observations, we show that this is in fact not the case. On the contrary, *Gandalf* can help in reducing the memory footprint while training, enabling researchers to use smaller GPUs, and reallocating their compute budget towards longer training schedules. Secondly, we also study the effect of subsampling the labels used for *Gandalf* to demonstrate how learning even some of the label-label correlations is beneficial for XMC models. This observation is particularly useful when inclusion of all label-features as data points becomes intractable due to its scale.

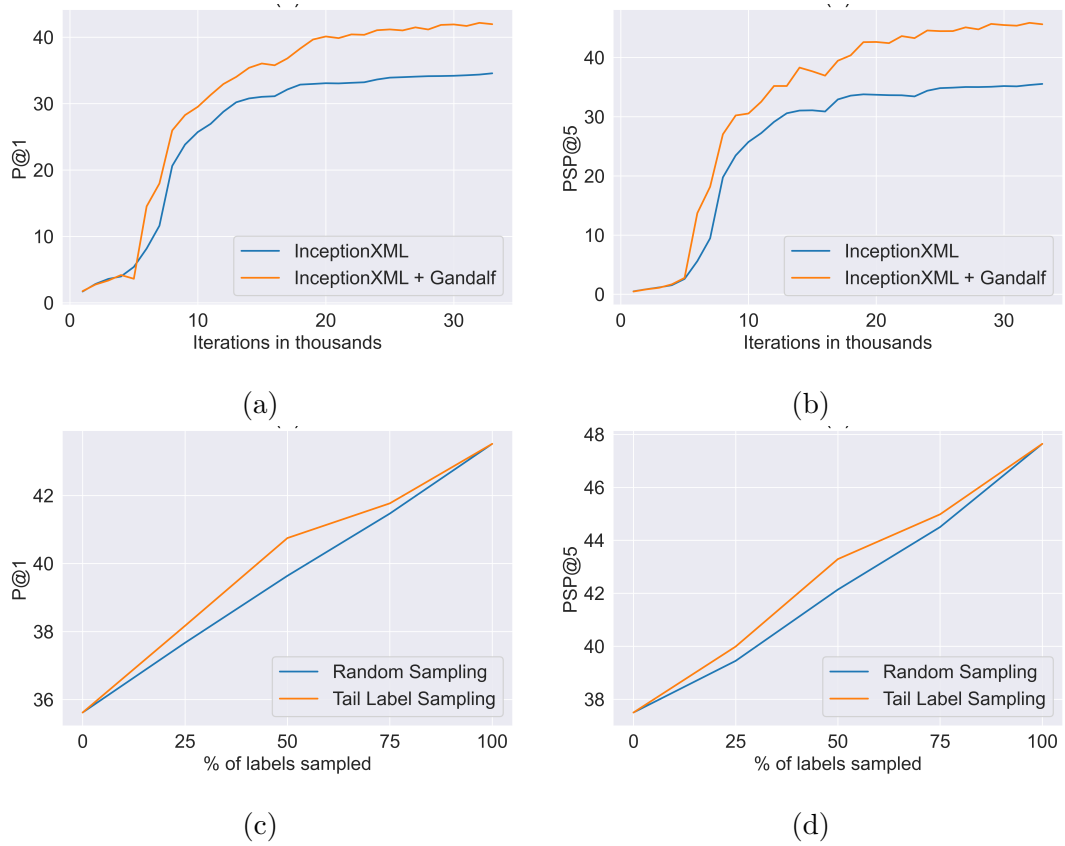


Figure 1.4: The (a) P@1 and (b) PSP@5 metric plotted against iterations for InceptionXML with and without *Gandalf*. The effect of subsampling labels for *Gandalf* on the (c) P@1 and (d) PSP@5 metric. Both results are on the LF-AmazonTitles-131K dataset.

Computational Costs during Training For the LF-Amazon- Titles-131K dataset, we plot the P@1 and the PSP@5 metric against iterations for InceptionXML, trained with and without Gandalf in Figure 1.4. As can be seen, using *Gandalf* gives better performance, even on tail labels, right from the beginning. Moreover, where the performance of InceptionXML saturates, the performance of *Gandalf* continues to scale with increasing compute. Therefore, given a fixed computational budget, a model trained with *Gandalf* will outperform one trained without it. This can also be seen in Table 1.2 where training on $\mathcal{U}(\mathcal{G}, N)$, i.e., under the exact same computational budget as training on the the original dataset gives performance improvements. In the same table, we can also observe improvements when training on *less than half* the original compute with $\mathcal{Z}$. These observations firmly place *Gandalf* as a compute-efficient method of leveraging label-features in XMC models.

Effect of Subsampling Labels We demonstrate the effect of subsampling labels used for *Gandalf* under two schemes, (a) Randomly sampling an expected percentage subset of labels and (b) randomly sampling this subset from equi-voluminous bins of increasing label frequency, i.e., prioritising tail labels for lower percentages. These results are shown for the P@1 and PSP@5 metric on the LF-AmazonTitles-131K dataset in Figure 1.4.

Both the metrics grow linearly as the percentage sampled labels are increased in steps of 25%. This goes ahead to show the lack of label-label correlations being captured in existing methods, and how learning even on a subset can be useful. Further, prioritising tail-labels consistently outperforms the random sampling baseline, underscoring the data-scarcity issue in XMC.

Comparison against conventional data augmentation strategies We compare Gandalf with with existing data augmentation techniques in Table 1.6. While no such techniques exist specifically for XMC, we use three baselines: synonym replacement(randomly replacing words in the input text with their synonyms, chosen via BERT similarity), MixUp and Label-MixUp. While the first two are standard data augmenta-

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
	LF-AmazonTitles-131K					
InceptionXML	35.62	24.13	17.35	27.53	33.06	37.50
+ Synonym Replacement	35.07	23.71	17.08	27.20	32.41	36.77
+ MixUp	35.63	24.15	17.37	27.55	33.00	37.63
+ Label-MixUp	37.25	25.02	17.98	29.25	34.58	39.09
+ Gandalf	**43.52**	**29.23**	**20.92**	**36.96**	**42.71**	**47.64**
	LF-WikiSeeAlsoTitles-320K					
InceptionXML	21.53	14.19	10.66	13.06	14.87	16.33
+ Synonym Replacement	20.08	13.13	9.92	12.00	13.50	14.90
+ MixUp	21.62	14.15	10.65	13.13	14.99	16.36
+ Label-MixUp	23.90	16.10	12.28	15.20	17.60	19.56
+ Gandalf	**31.31**	**21.38**	**16.22**	**24.31**	**26.79**	**28.83**

Table 1.6: Comparison of conventional data augmentation strategies with the proposed GANDALF method.

tions in NLP, Label-Mixup is a modified version of MixUp that combines the feature of a label feature and input datapoint, which is more suitable for XMC. Notably, Gandalf outperforms all of them with a significant margin:

Sensitivity to δ We examine *Gandalf*'s sensitivity to δ by training INCEPTIONXML-LF on data generated with varying values of δ. As shown in Table 1.7, the empirical performance peaks at a δ value of 0.1 which is sufficient to suppresses the impact of noisy correlations. Higher values of δ tend to suppress useful information.

Method	P@1	P@5	PSP@1	PSP@5	P@1	P@5	PSP@1	PSP@5
	LF-AmazonTitles-131K				**LF-WikiSeeAlsoTitles-320K**			
InceptionXML	35.62	17.35	27.53	37.50	21.53	10.66	13.06	16.33
+ *Gandalf* $\mathcal{G}$	**43.71**	**21.14**	**37.25**	**47.89**	**31.42**	**16.37**	**24.78**	**28.98**
+ *Gandalf* ($\mathcal{G}$ + *Random Walk*)	43.52	20.92	36.96	47.64	31.31	16.22	24.31	28.83
INCEPTIONXML-LF	40.74	19.57	34.52	44.13	49.01	39.46	24.56	31.67
+ *Gandalf* ($\delta = 0.0$)	41.71	20.14	36.94	46.64	31.40	16.53	26.01	29.99
+ *Gandalf* ($\delta = 0.1$)	**42.09**	**20.45**	**37.09**	**47.04**	**32.20**	**16.60**	**26.06**	**30.03**
+ *Gandalf* ($\delta = 0.2$)	41.73	20.18	37.01	46.67	31.29	16.28	25.68	29.65

Table 1.7: Results demonstrating the effectiveness of *Gandalf* using both, a statistical co-occurrence matrix ($\mathcal{G}$) and its modified version using a random walk. The table also shows the method's sensitivity to δ, as defined in Equation 1.6.

Choice of label co-occurrence graph $\mathcal{G}$ While with GANDALF, we leverage a statistical measure for $\mathcal{G}$, we can also estimate it with random walks (Mittal et al., 2021b). We find that our method is not significantly affected by this choice, with the co-occurrence graph giving slightly enhanced performance(Table 1.7). We hypothesise this happens due to the noise introduced via random walks. While both variants aim to model similar information, their differing usage determines their overall effectiveness. In particular, leveraging it for GANDALF helps learn sufficient information on top of *GALE*. Moreover, as discussed previously, our results are also not significantly affected by using the a symmetric variant of the graph, consistent with the GLaS regularizer, shown in Table 1.2.

1.6 Other Related Work

Prior works in XMC focused on annotating long-text documents, consisting of hundreds of word tokens, such as those encountered in tagging for Wikipedia (Babbar and Schölkopf, 2017; Khandagale et al., 2020; You et al., 2019; Schultheis and Babbar, 2022) with numeric label IDs. Most recent works under this setting were aimed towards scaling up transformer encoders for the XMC task (Zhang et al., 2021c; Kharbanda et al., 2022).

Exploiting Correlations in XMC For XMC datasets endowed with label features, there exist three correlations that can be exploited for better representation learning : (i) query-label, (ii) query-query, and (iii) label-label correlations. Recent works have been successful in leveraging label features and pushing state-of-the-art by exploiting the first two correlations. For example, SIAMESEXML and NGAME (Dahiya et al., 2021a, 2023a) employ a two-tower pre-training stage applying contrastive learning between an input text and its corresponding label features. GALAXC (Saini et al., 2021) & PINA (Chien et al., 2023), motivated by graph convolutional networks, create a combined query-label bipartite graph to aggregate predicted instance neighbourhood. This approach, however, leads to a multifold increase in the memory footprint. DECAF and ECLARE (Mittal et al., 2021a,b) make architectural additions to embed label-text embeddings (LTE) and

graph-augmented label embeddings (GALE) in each label's OVA classifier to exploit higher order correlations from the random walk graph. PINA, in its pre-training step, leverages label features as data points, but does so by expanding the label space $\{0,1\}^L$ to also include instances as $\{0,1\}^{L+N}$ leveraging the self-annotation property of labels (Dahiya et al., 2021a) and inverting the initial instance-label mappings to have instances $\mathbf{x}_i$ as labels for label features $\mathbf{z}_l$ as data points. This, however, leads to an explosion in an already enormous label space. In this work, we find that a significant amount of information can be learned by modelling label-label correlations, which existing methods fail to leverage.

Two-tower Models & Classifier Learning Typically, due to the single-annotation nature of most dense retrieval datasets (Nguyen et al., 2016; Kwiatkowski et al., 2019; Joshi et al., 2017), two-tower models (Karpukhin et al., 2020a) solving this task eliminate classifiers in favour of modelling implicit correlations by bringing query-document embeddings closer in the latent space of the encoders. These works are conventionally aimed at improving encoder representations by innovating on hard-negative mining (Zhang et al., 2021a; Xiong et al., 2020; Lu et al., 2022), teacher-model distillation (Qu et al., 2021; Ren et al., 2021) and combined dense-sparse training strategies (Khattab and Zaharia, 2020). While these approaches result in enhanced encoders, the multilabel nature of XMC makes them, in itself, insufficient for this domain. This has been demonstrated in two-stage XMC works like Dahiya et al. (2021a, 2023a); Jain et al. (2023) where these frameworks go beyond two-tower training and train classifiers with a frozen encoder in the second stage for better empirical performance. While a concurrent work (Gupta et al., 2023) does show that dual-encoder XMC models can outperform classifiers, but requires significant computational resources to scale the contrastive loss across the entire label space.

1.7 Conclusion

In this paper, we proposed *Gandalf*, a strategy to learn label correlations, a notoriously difficult challenge. In contrast to previous works which model these correlations implicitly through model training, we propose a supervised approach to explicitly learn them by leveraging the inherent query-label symmetry in short-text extreme classification. We further performed extensive experimentation by implementing on various SOTA XMC methods and demonstrated dramatic increases in prediction performances uniformly across all methods. Moreover, this is achieved with frugal architectures without incurring any computational overheads in inference latency or training memory footprint. We hope our treatment of label correlations in this domain will spur further research towards crafting data-points with more expressive annotations, and further extend it to long-text XMC approaches where the instance-label symmetry is quite ambiguous.

CHAPTER 2

UniDEC : Unified Dual Encoder and Classifier Training for Extreme Multi-Label Classification

2.1 Introduction

Extreme Multi-label Classification (XMC) is described as the task of identifying i.e. retrieving a subset, comprising of one or more labels, that are most relevant to the given data point from an extremely large label space, potentially consisting of millions of possible choices. Over time, XMC has increasingly found its relevance for solving multiple real world use cases. Typically, *long-text XMC* approaches are leveraged for the tasks of document tagging and product recommendation and *short-text XMC* approaches target tasks such as query-ad keyword matching and related query recommendation. Notably, in the real world manifestations of these use cases, the distribution of instances among labels exhibits a fit to Zipf's law (Adamic and Huberman, 2002). This implies, the vast label space ($L \approx 10^6$) is skewed and is characterized by the existence of *head*, *torso* and *tail* labels (Schultheis et al., 2022). For example, in query-ad keyword matching for search engines like Bing, Google etc. head keywords are often exact match or related phrase extensions of popularly searched queries while tail keywords often target specific niche queries. Typically, we can characterize head, torso and tail keywords as having > 100, 10 - 100, and 1 - 10 annotations, respectively.

Typically, per-label classifiers are employed for solving the XMC task. A naive strategy to train classifiers for XMC involves calculating the loss over the entire label set. This method has seen empirical success, particularly in earlier works like DiSMEC (Babbar and Schölkopf, 2017), which learns one-vs-all classifiers by parallelisation across

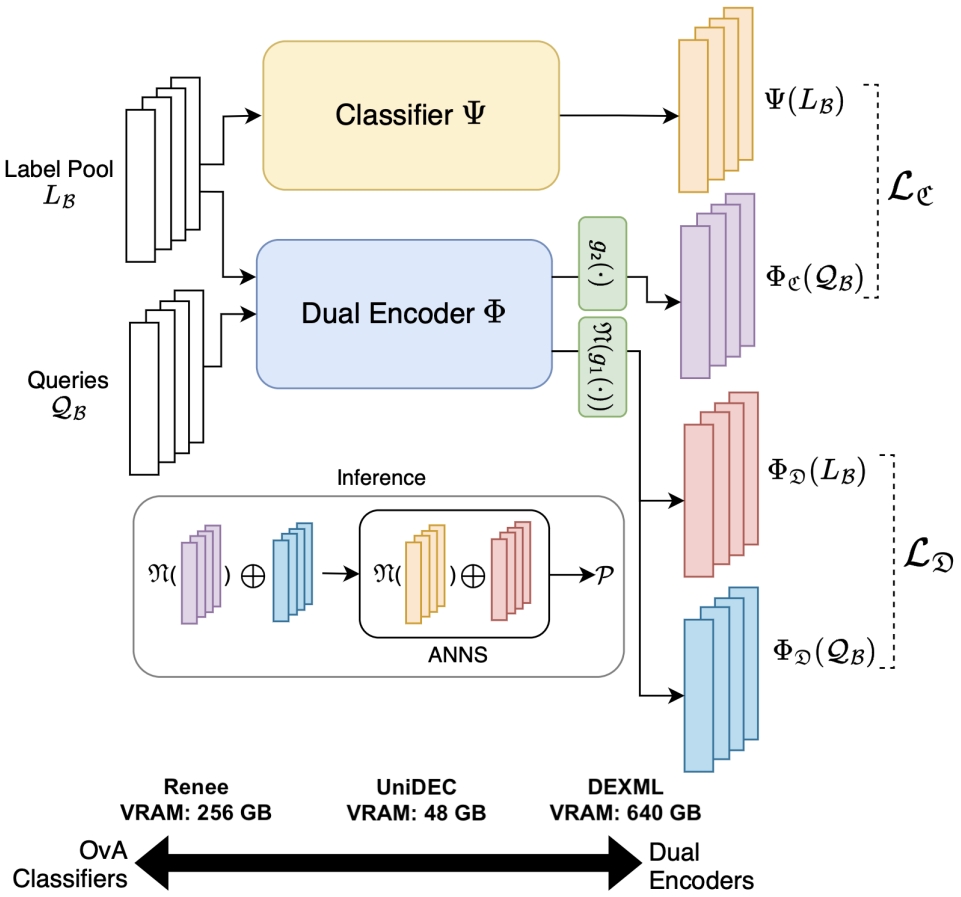


Figure 2.1: The architecture for the UniDEC framework, denoting the the classifiers and DE trained in parallel, along with the loss functions used. The inference pipeline is shown in the rectangular box.

multiple CPU cores. With the adoption of deep encoders, various works moved to training on GPUs, which due to VRAM constraints, led to the use of tree-based shortlisting strategies (Chang et al., 2020; Kharbanda et al., 2022, 2023; Dahiya et al., 2021b) to train classifiers on the hardest negatives. This reduced the computational complexity from $\mathcal{O}(L)$ to $\mathcal{O}(\log L)$. While leveraging a label shortlist made XMC training more feasible via modular training (Dahiya et al., 2023a,b) or joint training using a meta-classifier (Jiang et al., 2021; Kharbanda et al., 2022, 2023) , it still left out scope for empirical improvements. Consequently, Renee (Jain et al., 2023) demonstrated the extreme case for methods which employ classifiers with deep encoders by writing custom CUDA kernels to scale classifier training over the entire label space. This, however, leads to a GPU VRAM usage of 256GB (Figure 2.1) for training a DistilBERT model on

LF-AmazonTitles-1.3M, the largest XMC dataset. Notably, what remains common across all XMC classifier-training algorithms is the advocacy of OvA reduction for the multi-label problem (Dahiya et al., 2021a). Theoretically, the alternate pick-all-labels (PAL) approach should lead to a better optimization over OvA, since it promotes "competition" amongst labels (Menon et al., 2019). However, PAL reduction has neither been well-studied nor successfully leveraged to train classifiers in XMC since such losses require considering all labels which is prohibitively expensive.

A parallel line of research involves leveraging dual encoders (DE) for XMC. While DE models are a popular choice for dense retrieval (DR) and open-domain question answering (ODQA) tasks, these are predominantly *few* and *zero-shot* scenarios. In contrast, XMC covers a broader range of scenarios (see Table 1.1). Consequently, modelling the XMC task as a retrieval problem is tantamount to training a DE simultaneously on *many*, *few* and *one-shot* scenarios. While DE trained with triplet loss was thought to be insufficient for XMC, and thus augmented with per-label classifiers to enhance performance (Dahiya et al., 2023a; Gupta et al., 2023), a recent work DEXML (Gupta et al., 2023) proved the sufficiency of the DE framework for XMC by proposing a new multi-class loss function *Decoupled Softmax*, which computed the loss over the entire label space. This, however, turns out to be prohibitively expensive as DEXML requires 640GB GPU VRAM to train on LF-AmazonTitles-1.3M.

At face value, PAL reduction of multi-label problems for DE training should be made tractable by optimizing over in-batch labels, however in practice, it does not scale to larger datasets due to the higher number of positives per label. For instance, for LF-AmazonTitles-1.3M a batch consisting of 1,000 queries will need an inordinately large label pool of size $\sim$ 22.2K (considering in-batch negatives) to effectively train a DE with the PAL loss. Alternatively, the stochastic implementation of PAL in the form of pick-one-label (POL) reduction used by DEXML, either convergences slowly (Gupta et al., 2023) or fails to reach SOTA performance.

In order to enable efficient training, in this work, we propose ***"pick-some-labels"*** *(PSL)* relaxation of the PAL reduction for the multi-label classification problem which

enables scaling to large datasets ($\sim 10^6$ labels). Here, instead of trying to include all the positive labels for instances in a batch, we propose to randomly sample at max β positive labels per instance. To the best of our knowledge, we are the first work to study the effect of multi-class losses for training classifiers at an extreme scale. Further, we aim to develop an end-to-end trainable loss-independent framework, UniDEC - **Uni**fied **D**ual **E**ncoder and **C**lassifier, for XMC that leverages the multi-positive nature of the XMC task to create highly informative in-batch labels to train the DE, and be used as a shortlist for the classifier. As shown in Figure 2.1, UniDec, in a single pass, performs an update step over the combined loss computed over two heads: (i) between DE head's query and sampled label-text embeddings, (ii) between CLF head's query embeddings and classifier weights corresponding to sampled labels. By unifying the two compute-heavy ends of the XMC spectrum in such a way, UniDEC is able to significantly reduce the training computational cost down to a **single 48GB GPU**, even for the largest dataset with 1.3M labels. End-to-end training offers multiple benefits as it (i) helps us do away with a meta-classifier and modular training, (ii) dynamically provides progressively harder negatives with lower GPU VRAM consumption, which has been shown to outperform static negative mining (Jiang et al., 2021; Kharbanda et al., 2022, 2023) (iii) additionally, with an Approximate Nearest Neighbour Search (ANNS), it can explicitly mine hard negative labels added to the in-batch negatives. While UniDEC is a loss independent framework (see Table 2.4), the focus of this work also includes studying the use of multi-class losses for training multi-label classifiers at an extreme scale via the proposed *PSL* reduction. To this end, we benchmark UniDEC on 6 public datasets, forwarding the state-of-the-art in each, and a proprietary dataset containing 450M labels. Finally, we also experimentally show how OvA losses like BCE can be applied in tandem with multi-class losses for classifier training.

2.2 Related Works & Preliminaries

For training, we have available a multi-label dataset $\mathcal{D} = \{\{\mathbf{x}_i, \mathcal{P}_i\}_{i=1}^N, \{\mathbf{z}_l\}_{l=1}^L\}$ comprising of N data points and L labels. Each $\mathbf{x}_i$ is associated with a small ground truth label set $\mathcal{P}_i \subset [L]$ out of $L \sim 10^6$ possible labels. Further, $\mathbf{x}_i, \mathbf{z}_l \in \mathcal{X}$ denote the textual descriptions of the data point i and the label l respectively, which, in this setting, derive from the same vocabulary universe $\mathcal{V}$ (Dahiya et al., 2021a). The goal is to learn a parameterized function f which maps each instance $\mathbf{x}_i$ to the vector of its true labels $\mathbf{y}_i \in [0, 1]^L$ where $\mathbf{y}_{i,l} = 1 \Leftrightarrow l \in \mathcal{P}_i$.

Dual Encoder A dual encoder consists of the query encoder Φ_q, and a label encoder Φ_l. Conventionally, the parameters for Φ_q and Φ_l are shared, and thus we will simply represent it as Φ (Dahiya et al., 2023a; Gupta et al., 2023; Karpukhin et al., 2020b; Qu et al., 2021). The mapping $\Phi(.)$ projects the instance $\mathbf{x}_i$ and label-text $\mathbf{z}_l$ into a shared d-dimensional unit hypersphere $\mathcal{S}^{d-1}$. For each instance $\mathbf{x}_i$, its similarity with label $\mathbf{z}_l$ is then computed via an inner product i.e., $s_{i,l} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{z}_l) \rangle$ to produce a ranked list of top-K relevant labels.

Training two-tower algorithms for XMC at scale is made possible by recursively splitting (say, via a hierarchical clustering strategy) instance encoder embeddings $\{\Phi(\mathbf{x}_i)\}_{i=1}^N$ into disjoint clusters $\mathfrak{B}$ (Dahiya et al., 2023a), where each cluster represents a training batch $\mathcal{B}$. Each batch $\mathcal{B} = \{Q_\mathcal{B}, L_\mathcal{B}\}$ is characterised by a set of instance indices $Q_\mathcal{B} = \{i \mid i \in [N]\},\ s.t.\ |Q_\mathcal{B}| = N/|\mathfrak{B}|$, and the corresponding collated set of (typically one per instance) sampled positive labels $p \in \mathcal{P}_i$, defined as $L_\mathcal{B} = \{p \mid p \in \mathcal{P}_i \text{ and } i \in Q_\mathcal{B}\}$. As per the in-batch negative sampling strategy common across existing works (Dahiya et al., 2023a; Gupta et al., 2023; Karpukhin et al., 2020b; Qu et al., 2021), the negative label pool then is made up of the positive labels sampled for other instances in the batch i.e. $\mathcal{N}_i = L_\mathcal{B} - \mathcal{P}_i$. As compared to random batching, (Dahiya et al., 2023a) posit that the batches created from instance-clustering are *negative-mining aware* i.e. for every instance, the sampled positives of the other instances in the batch serve as the set of appropriate "hard" negatives.

An additional effect of this is the accumulation of multiple in-batch positives for most queries (see Figure 2.2a). This makes the direct application of commonly used multi-class loss - InfoNCE loss - infeasible for training DE. Hence XMC methods find it suitable to replace InfoNCE loss with a triplet loss (Dahiya et al., 2023a,b) or probabilistic contrastive loss (Dahiya et al., 2021a), as it can be potentially applied over multiple positives and hard negatives (equation 1 in (Dahiya et al., 2023a)). While this would seem favourable, these approaches still fail to leverage the additional positive signals owing to multiple positives in the batch as they calculate loss over only a single sampled positive i.e. employing POL reduction instead of PAL reduction.

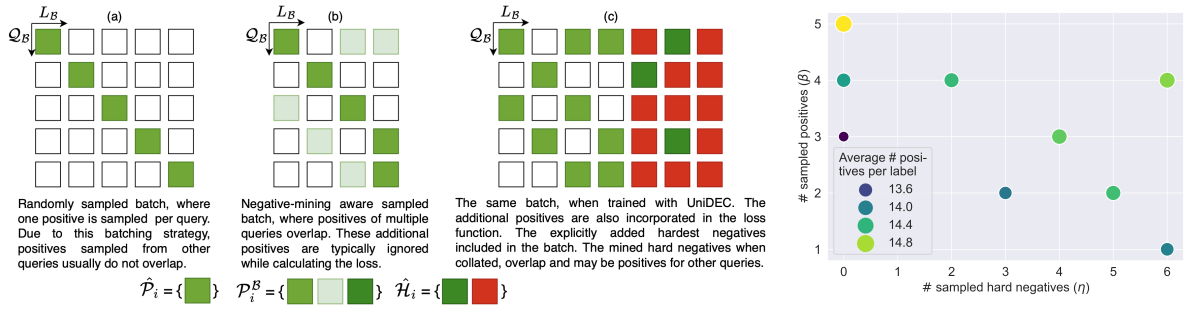
Classifiers in XMC The traditional XMC set-up considers labels as featureless integer identifiers which replace the encoder representation of labels $\Phi(\mathbf{z}_l)$ with learnable classifier embeddings $\Psi_{l=1}^{L} \in \mathbb{R}^{L \times d}$ (Chang et al., 2020; You et al., 2019; Zhang et al., 2021b). The relevance of a label l to an instance is scored using an inner product, $s_{i,l} = \langle \Phi(\mathbf{x}_i), \Psi_l \rangle$ to select the k highest-scoring labels. Under the conventional OvA paradigm, each label is independently treated with a binary loss function ℓ_{BC} applied to each entry in the score vector. The OvA reduction can be expressed as,

$$\mathcal{L}_{\text{OVA}} = \sum_{l=1}^{L} \{ y_l \cdot \ell_{BC}(1, s_{i,l}) + (1 - y_l) \cdot \ell_{BC}(0, s_{i,l}) \}$$

2.3 Method: UniDEC

In this work, we propose a novel multi-task learning framework which, in an end-to-end manner, trains both - a dual encoder and extreme classifiers - in parallel. The framework eliminates the need of a meta classifier for a dynamic in-batch shortlist. Further, it provides the encoder with the capability to explicitly mine hard-negatives, obtained by querying an ANNS, created over $\{\Phi(\mathbf{z}_l)\}_{l=1}^{L}$, which is refreshed every ε epochs.

The DE head is denoted by $\Phi_{\mathfrak{D}}(\cdot) = \mathfrak{N}(g_1(\Phi(\cdot)))$ and the classifier head by $\Phi_{\mathfrak{C}}(\cdot) = g_2(\Phi(\cdot))$, where $\mathfrak{N}$ represents the L2 normalization operator and $g_1(\cdot)$ and $g_2(\cdot)$ represent separate nonlinear projections. Unlike DE, and as is standard practice for OvA classifiers,



(a) (b)

Figure 2.2: (a) Visualizing UniDEC's batching strategy. Such a framework naturally leads to higher number of positives per query, enabling us to scale without increasing the batch size significantly. (b) Scatter plot showing the average number of positive labels per query, when we sample β positives and η hard negatives in the batch. Note that, even with $\beta = 3$ and $\eta = 0$, avg($|P|$) = 13.6.

we train them without an additional normalization operator (Dahiya et al., 2021a,b).

2.3.1 *Pick-some-Labels* Reduction

$\mathcal{L}_{\text{PAL-N}}$ (Menon et al., 2019) is is formulated as :

$$\mathcal{L}_{\text{PAL-N}}(\Phi_1(\mathbf{x}), \Phi_2(z_l)) = \frac{1}{\sum_{j=1}^{L} y_j} \sum_{l=1}^{L} y_l \cdot \ell_{MC}(1, \langle \Phi_1(\mathbf{x}), \Phi_2(z_l) \rangle)$$

Since it computes the loss over the entire label space, it is computationally intractable for XMC scenarios. To reduce the computational costs associated with this reduction, we propose a relaxation by computing loss over some labels in batch $\mathcal{B} = \{Q_\mathcal{B}, L_\mathcal{B}\}$, which we call *pick-some-labels (PSL)*.

$$\mathcal{L}_{PSL}(\Phi_1(\mathbf{x}), \Phi_2(z_l) \mid \mathcal{B}, \mathcal{P}^\mathcal{B}) = \sum_{i \in Q_\mathcal{B}} \frac{-1}{|\mathcal{P}_i^\mathcal{B}|} \sum_{p \in \mathcal{P}_i^\mathcal{B}} \ell_{MC}(1, \langle \Phi_1(\mathbf{x}), \Phi_2(z_p) \rangle)$$

where Φ_1 and Φ_2 are encoding networks. Any multi-class loss [2] can be used in place of ℓ_{MC}. By varying Φ_1 and Φ_2, we get a generic loss function for training classifier as well

[2]While binary class loss functions can also be used, in this work, our focus is to study multi-class losses

as DE. This approximation enables employing PAL-N over a minibatch $Q_{\mathcal{B}}$ by sampling a subset of positive labels $\hat{\mathcal{P}}_i \subseteq \mathcal{P}_i \; s.t. \; |\hat{\mathcal{P}}_i| \leq \beta$. The collated label pool, considering in-batch negative mining, is defined as $L_{\mathcal{B}} = \{\bigcup_{i \in Q_{\mathcal{B}}} \hat{\mathcal{P}}_i\}$. Here, $\mathcal{P}_i^{\mathcal{B}} = \{\mathcal{P}_i \cap L_{\mathcal{B}}\}$ denotes all the in-batch positives for an instance $\mathbf{x}_i$, i.e., the green and pale green in Figure 2.2.

2.3.2 Dual Encoder Training with *Pick-some-Labels*

The PSL loss to train a DE is formulated as,

$$\mathcal{L}_{\mathfrak{D},q2l} = \mathcal{L}_{\text{PSL}}(\Phi_{\mathfrak{D}}(\mathbf{x}), \Phi_{\mathfrak{D}}(z_l) \mid \mathcal{B}, \mathcal{P}^{\mathcal{B}})$$

More specifically, we perform k-means clustering on the queries such that similar queries are clustered into the same batch $\mathfrak{B}$, leading to both positive and negative-aware batching (Dahiya et al., 2023a). Thus, $\mathcal{P}_i^{\mathcal{B}}$ consists not only of the sampled positives $\hat{\mathcal{P}}_i$ but also those non-sampled positives that exist in the batch as sampled positives of other instances i.e. $\mathcal{P}_i^{\mathcal{B}} = \hat{\mathcal{P}}_i \; \cup \; \{\bigcup_{j \in \{Q_{\mathcal{B}} - \{i\}\}} \hat{\mathcal{P}}_j \; \cap \; \mathcal{P}_i\}$. We find the cardinality of the second term to be non-zero for most instances having $|\mathcal{P}_i| > \beta$ due to a high overlap of sampled positive labels in query-clustered batches, leading to a more optimal batch size. Thus, although we sample $|\hat{\mathcal{P}}_i| \leq \beta \; \forall \; i \in Q_{\mathcal{B}}, \; \exists \; i \in Q_{\mathcal{B}} \; s.t. \; |\mathcal{P}_i^{\mathcal{B}}| \geq \beta$. As per our observations, $\mathcal{P}_i^{\mathcal{B}} = \mathcal{P}_i$ for most tail and torso queries. For e.g., even if $\beta = 1$ for LF-AmazonTitles-1.3M, for $|Q_{\mathcal{B}}| = 10^3, \; \text{Avg}(|\hat{\mathcal{P}}_i|) = [12, 14]$. Thus, it makes *PSL* reduction same as PAL for torso and tail labels and only taking form of *PSL* for head queries.

Dynamic ANNS Hard-Negative Mining While the above strategy leads to collation of hard negatives in a batch, it might not mine hardest-to-classify negatives (Dahiya et al., 2023a). We explicitly add them by querying an ANNS created over $\{\Phi_{\mathfrak{D}}(\mathbf{z}_l)\}_{l=1}^{L}$ for all $\{\Phi_{\mathfrak{D}}(\mathbf{x}_i)\}_{i=1}^{N}$. More specifically, for each instance, we create a list of hard negatives $\mathcal{H}_i = \text{top}_{\text{k}}(\text{ANNS}(\Phi_{\mathfrak{D}}(\mathbf{x}_i)|_{i=1}^{N}, \; \Phi_{\mathfrak{D}}(\mathbf{z}_l)|_{l=1}^{L})) \; s.t. \; \mathcal{H}_i \cap \mathcal{P}_i = \phi$ (denoted by red in Figure 2.2). Every iteration, we uniformly sample a η-sized hard-negative label subset $\hat{\mathcal{H}}_i \subset \mathcal{H}_i$ alongside $\hat{\mathcal{P}}_i \; \forall \; \mathbf{x}_i \in Q_{\mathcal{B}}$. More formally, the new batch label pool can be denoted

as $L_{\mathcal{B}} = \{\bigcup_{i \in Q_{\mathcal{B}}} \hat{\mathcal{P}}_i \ \cup \ \hat{\mathcal{H}}_i\}$. Interestingly, due to the multi-positive nature of XMC, sampled hard-negatives for $\mathbf{x}_i$ might turn out to be an unsampled positive label for $\mathbf{x}_j$. More formally, $\exists\ j \in Q_{\mathcal{B}}\ s.t.\ \{\hat{\mathcal{H}}_i \ \cap \ \mathcal{P}_j \neq \phi,\ \hat{\mathcal{H}}_i \ \cap \ \mathcal{P}_j^{\mathcal{B}} = \phi\}$. This requires altering the definition of $\mathcal{P}_i^{\mathcal{B}}$ to accommodate these *extra* positives (represented by the dark green square in Figure 2.2) as $\mathcal{P}_i^{\mathcal{B}} = \{\hat{\mathcal{P}}_i \ \cup \ \{\bigcup_{j \in \{Q_{\mathcal{B}} - \{i\}\}} \{\hat{\mathcal{P}}_j \cup \hat{\mathcal{H}}_j\} \ \cap \ \mathcal{P}_i\}\}$. This effect is also quantified in Figure 2.2b. Query clustering for batching and dynamic ANNS hard-negative mining strategies complement each other, since the presence of similar queries leads to a higher overlap in their positives *and hard negatives*, enabling us to scale the effective size of the label pool. Further, to provide $\Phi_{\mathfrak{D}}$ and $\Phi_{\mathfrak{C}}$ with progressively harder negatives, the ANNS is refreshed every τ epochs and to uniformly sample hard negatives, we keep $|\mathcal{H}| = \eta \times \tau$.

Note that $L_{\mathfrak{D},q2l}$ denotes the multi-class loss between $\mathbf{x}_i$ and $\mathbf{z}_l\ \forall\ l \in L_{\mathcal{B}}$. As the data points and labels in XMC tasks belong to the same vocabulary universe (such as product recommendation), we find it beneficial to optimize $L_{\mathfrak{D},l2q}$ alongside $L_{\mathfrak{D},q2l}$, making $L_{\mathfrak{D}}$ a symmetric loss. Since (Radford et al., 2021), a plethora of works have leveraged symmetric optimizations in the vision-language retrieval pre-training domain. For XMC, the interchangability of $Q_{\mathcal{B}}$ and $L_{\mathcal{B}}$ in the symmetric objective can be viewed equivalent to (i) feeding more data relations in a batch, and (ii) bridging missing relations in the dataset. Further, we formulate XMC as a symmetric problem from $L_{\mathcal{B}}$ to $Q_{\mathcal{B}}$, thus calculating the multi-class loss between $\mathbf{z}_l$ and $\mathbf{x}_i\ \forall\ i \in Q_{\mathcal{B}}$ given by:

$$\mathcal{L}_{\mathfrak{D},l2q} = \mathcal{L}_{\text{PSL}}(\Phi_{\mathfrak{D}}(z_l), \Phi_{\mathfrak{D}}(\mathbf{x}) \mid \mathcal{B}, \mathcal{P}^L)$$

Note that, $\mathcal{P}^L = \{i \mid i \in Q_{\mathcal{B}},\ l \in L_{\mathcal{B}},\ \mathcal{P}_{i,l} = 1\}$. The total DE contrastive loss can thus be written as (note, for simplicity we use $\lambda_{\mathfrak{D}} = 0.5$ for all datasets, which works well in practice):

$$\mathcal{L}_{\mathfrak{D}} = \lambda_{\mathfrak{D}} \cdot \mathcal{L}_{\mathfrak{D},q2l} + (1 - \lambda_{\mathfrak{D}}) \cdot \mathcal{L}_{\mathfrak{D},l2q}$$

Algorithm 1 Training step in UniDEC

Input: instance $\mathbf{x}$, label features $\mathbf{z}$, positive labels $\mathcal{P}$, encoder Φ, classifier lookup-table Ψ, non-linear transformations $g1(\cdot)$ and $g2(\cdot)$

$\Phi_{\mathfrak{D}}(\cdot),\ \Phi_{\mathfrak{C}}(\cdot) \coloneqq \mathfrak{N}(g_1(\Phi(\cdot))),\ g_2(\Phi(\cdot))$

for e *in* $1..\epsilon$ **do**

  if $e\ \%\ \tau$ *is* 0 **then**

    $\mathfrak{B} \leftarrow \textsc{Cluster}(\Phi_{\mathfrak{D}}(\mathbf{x}_i)|_{i=0}^{N})$

    $\mathcal{H} \leftarrow \text{topk}(\text{ANNS}(\Phi_{\mathfrak{D}}(\mathbf{x}_i)|_{i=0}^{N},\ \Phi_{\mathfrak{D}}(\mathbf{z}_l)|_{l=0}^{L}))$

  for $Q_{\mathcal{B}}$ *in* $\mathfrak{B}$ **do**

    for i *in* $Q_{\mathcal{B}}$ **do**

      $\hat{\mathcal{P}}_i \leftarrow \text{sample}(\mathcal{P}_i, \beta)$

      $\hat{\mathcal{H}}_i \leftarrow \text{sample}(\mathcal{H}_i - \mathcal{P}_i, \eta)$

    $L_{\mathcal{B}} \leftarrow \{\bigcup_{i \in Q_{\mathcal{B}}} \hat{\mathcal{P}}_i\ \cup\ \hat{\mathcal{H}}_i\}$

    $\mathcal{P}^{\mathcal{B}} \leftarrow \{\{\mathcal{P}_i \cap L_{\mathcal{B}}\}|_{i \in Q_{\mathcal{B}}}\}$

    $\mathcal{P}^{L} \leftarrow \{\{i \mid i \in Q_{\mathcal{B}},\ \mathcal{P}_{i,l} = 1\}|_{l \in L_{\mathcal{B}}}\}$

    $\mathcal{L}_{\mathfrak{D},q2l} \leftarrow \mathcal{L}_{PSL}(\Phi_{\mathfrak{D}}(\mathbf{x}_i), \Phi_{\mathfrak{D}}(\mathbf{z}_l) \mid \mathcal{B},\ \mathcal{P}^{\mathcal{B}})$

    $\mathcal{L}_{\mathfrak{D},l2q} \leftarrow \mathcal{L}_{PSL}(\Phi_{\mathfrak{D}}(\mathbf{z}_l), \Phi_{\mathfrak{D}}(\mathbf{x}_i) \mid \mathcal{B},\ \mathcal{P}^{L})$

    $\mathcal{L}_{\mathfrak{D}} \leftarrow \lambda_{\mathfrak{D}} \cdot \mathcal{L}_{\mathfrak{D},q2l} + (1 - \lambda_{\mathfrak{D}}) \cdot \mathcal{L}_{\mathfrak{D},l2q}$

    $\mathcal{L}_{\mathfrak{C},q2l} \leftarrow \mathcal{L}_{PSL}(\Phi_{\mathfrak{C}}(\mathbf{x}_i), \Psi(l) \mid \mathcal{B},\ \mathcal{P}^{\mathcal{B}})$

    $\mathcal{L}_{\mathfrak{C},l2q} \leftarrow \mathcal{L}_{PSL}(\Psi(l), \Phi_{\mathfrak{C}}(\mathbf{x}_i) \mid \mathcal{B},\ \mathcal{P}^{L})$

    $\mathcal{L}_{\mathfrak{C}} \leftarrow \lambda_{\mathfrak{C}} \cdot \mathcal{L}_{\mathfrak{C},q2l} + (1 - \lambda_{\mathfrak{C}}) \cdot \mathcal{L}_{\mathfrak{C},l2q}$

    $\mathcal{L} \leftarrow \lambda \cdot \mathcal{L}_{\mathfrak{D}}\ +\ (1 - \lambda) \cdot \mathcal{L}_{\mathfrak{C}}$

  adjust Φ, $g_1(\cdot)$, $g_2(\cdot)$ and Ψ to reduce loss $\mathcal{L}$.

2.3.3 Unified Classifier Training with *Pick-some-Labels*

XMC classifiers are typically trained on a shortlist consisting of *all* positive and $\mathcal{O}(Log(L))$ hard negative labels (Dahiya et al., 2021b). As the reader can observe from Figure 2.1 and Algorithm 1, the document and label embedding computation and batch pool is shared between $\Phi_{\mathfrak{D}}$ and $\Phi_{\mathfrak{C}}$. We simply unify the classifier training with that of DE by

leveraging the same *PSL* reduction used for contrastive learning, with only minor changes: $\Phi_{\mathfrak{C}}(\mathbf{x}_i)|_{i \in Q_{\mathcal{B}}}$ replaces $\Phi_{\mathfrak{D}}(\mathbf{x}_i)|_{i \in Q_{\mathcal{B}}}$ and, the label embeddings $\Phi_{\mathfrak{D}}(\mathbf{z}_l)|_{l \in L_{\mathcal{B}}}$ are replaced by $\Psi_l|_{l \in L_{\mathcal{B}}}$. Formally, the multi-class *PSL* loss for classifier $L_{\mathfrak{C},q2l}$ can be defined as:

$$\mathcal{L}_{\mathfrak{C},q2l} = \mathcal{L}_{\text{PSL}}(\Phi_{\mathfrak{C}}(\mathbf{x}), \Psi_l \mid \mathcal{B}, \mathcal{P}^{\mathcal{B}})$$

Similar to DE training, we find it beneficial to employ a symmetric loss for classifier training as well, defined (with $\lambda_{\mathfrak{C}} = 0.5$) as:

$$\mathcal{L}_{\mathfrak{C}} = \lambda_{\mathfrak{C}} \cdot \mathcal{L}_{\mathfrak{C},q2l} + (1 - \lambda_{\mathfrak{C}}) \cdot \mathcal{L}_{\mathfrak{C},l2q}$$

Finally, we combine the two losses and train together in an end-to-end fashion, thereby achieving Unification of DE and classifier training for XMC.

$$\mathcal{L} = \lambda \mathcal{L}_{\mathfrak{D}} + (1 - \lambda)\mathcal{L}_{\mathfrak{C}}$$

2.3.4 Inference

For ANNS inference, the label graph can either be created over the encoded label embeddings $\{\Phi_{\mathfrak{D}}(z_l)\}_{l=1}^{L}$ or the label classifier embeddings $\{\mathfrak{N}(\Psi(l))\}_{l=1}^{L}$, which are queried by $\{\Phi_{\mathfrak{D}}(\mathbf{x}_i)\}_{i=1}^{N}$ or $\{\mathfrak{N}(\Phi_{\mathfrak{C}}(\mathbf{x}_i))\}_{i=1}^{N}$ respectively. Even though we train the classifiers over an un-normalized embedding space, we find it empirically beneficial to perform ANNS search over the unit normalized embedding space (Gunel et al., 2021; Khosla et al., 2020). Interestingly, the concatenation of these two embeddings leads to a much more efficient retrieval. More specifically, we create the ANNS retrieval graph over the concatenated label representation $\{\Phi_{\mathfrak{D}}(z_l) \oplus \mathfrak{N}(\Psi(l))\}|_{l=0}^{L}$, which is queried by the concatenated document representations $\{\Phi_{\mathfrak{D}}(\mathbf{x}_i) \oplus \mathfrak{N}(\Phi_{\mathfrak{C}}(\mathbf{x}_i))\}|_{i=0}^{N}$. Intuitively, this is a straight-forward way to ensemble the similarity scores from both the embedding spaces.

2.4 Experiments

Datasets: We benchmark our experiments on 6 standard datasets, comprising of both long-text inputs (LF-Amazon-131K, LF-WikiSeeAlso-320K) and short-text inputs (LF-AmazonTitles-131K, LF-AmazonTitles-1.3M, LF-WikiTitles-500K, LF-WikiSeeAlsoTitles-320K). We also evaluate baselines on a proprietary *Query2Bid* dataset, comprising of 450M labels, which is orders of magnitude larger than any public dataset. Details of these datasets can be found at (Bhatia et al., 2016) and in Table 2.1.

Datasets	Benchmark	N	L	APpL	ALpP	AWpP
MS-MARCO	DR	502,931	8,841,823	-	1.1	56.58
LF-AmazonTitles-131K	XMC	294,805	131,073	5.15	2.29	6.92
LF-Amazon-131K	XMC	294,805	131,073	5.15	2.29	6.92
LF-AmazonTitles-1.3M	XMC	2,248,619	1,305,265	38.24	**22.20**	8.74
LF-WikiSeeAlso-320K	XMC	693,082	312,330	4.67	2.11	3.01
Query2Bid-450M	Search Engine	52,029,024	454,608,650	34.61	3.96	-

Table 2.1: Details of the benchmark datasets with label features. APpL stands for avg. points per label, ALpP stands for avg. labels per point and AWpP is the length i.e. avg. words per point.

MS-MARCO, a representative dataset for DR tasks, has 3.2M documents but on average contains only 1.1 positively annotated answers (label) per question (instance) (Qu et al., 2021). On the other hand, LF-AmazonTitles-1.3M, an XMC dataset which is representative dataset for product recommendation task, has a label space spanning 1.3M Amazon products where each instance (a product title) is annotated (tagged), by ~ 22.2 labels (related product titles) and each label annotates, ~ 38.2 instances. This indicates the broader spectrum of XMC tasks in contrast with zero-shot nature of ODQA task.

Baselines & Evaluation Metrics: We compare against two classes of Baselines namely, (i) DE Approaches (Φ) consisting of only an encoder (Dahiya et al., 2023a; Gupta et al., 2023; Karpukhin et al., 2020b; Xiong et al., 2020) and, (ii) Classifier Based Approaches (Ψ) which use linear classifiers, with or without the encoder (Dahiya et al., 2023a; Jain et al., 2023). We use popular metrics such as Precision@K and Propensity-scored Precision@K (K $\in \{1, 3, 5\}$), defined in (Bhatia et al., 2016).

Implementation Details We initialize both DePSL, our purely dual encoder method

and UniDEC with a pre-trained 6L-DistilBERT and train the Φ, $g(\cdot)$ and Ψ with a learning rate of $1e-4$, $2e-4$ and $1e-3$ respectively using cosine annealing with warm-up as the scheduler, hard-negative shortlist refreshed every $\tau = 5$ epochs. We make an effort to minimize the role of hyperparameters by keeping them almost same across all datasets. Note that for all experiments in the paper, $g_1(\cdot)$, $g_2(\cdot)$ is defined as follows,

$$\Phi_{\mathfrak{D}}(\cdot),\ \Phi_{\mathfrak{C}}(\cdot) \coloneqq \mathfrak{N}(g_1(\Phi(\cdot))),\ g_2(\Phi(\cdot))$$
$$g_1(\cdot) \coloneqq \text{nn.Sequential}(\text{nn.Linear}(\text{d}_\Phi, \text{d}), \text{nn.Tanh}(), \text{nn.Dropout}(0.1))$$
$$g_2(\cdot) \coloneqq \text{nn.Sequential}(\text{nn.Linear}(\text{d}_\Phi, \text{d}), \text{nn.Dropout}(0.1))$$

2.4.1 Evaluation Results

In these settings, we evaluate DePSL and UniDEC against both DE and XMC baselines. UniDEC differs from these baselines in the following ways, (i) on training objective, UniDEC uses the proposed *PSL* relaxation of PAL for both DE and CLF training, instead of POL reduction used by existing methods like Ngame and Dexml, (ii) UniDEC does away with the need of modular training by unifying DE and CLF, (iii) finally, UniDEC framework adds explicitly mined hard negatives to the negative mining-aware batches which helps increase P@K metrics (see Table 2.5). Note that direct comparison against some of the baselines may not be justifiable due to reasons discussed individually below.

UniDEC/DePSL vs Ngame(Φ): Table 2.2 depicts that UniDEC ($\Phi \oplus \Psi$) consistently outperforms Ngame(Ψ) (it's direct comparison baseline), where we see gains of $2-8\%$ in P@K and upto 10% on PSP@K. DePSL, on the other hand, outperforms Ngame on P@k with improvements ranging from $2-9\%$. For PSP@k, DePSL (Φ) always outperforms Ngame (Φ) on long-text datasets, while the results are mixed on short-text datasets.

DePSL vs DPR/ANCE: Empirical performance of DPR demonstrates the limits of a DE model trained with InfoNCE loss and random in-batch negatives (popular in DR methods). Evidently, Ance improves over DPR in the P@K metrics, which can be

Method	d	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	d	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
Long-text →		LF-Amazon-131K							LF-WikiSeeAlso-320K					
NGAME (Φ)	768	42.61	28.86	20.69	**38.27**	43.75	48.71	768	43.58	28.01	20.86	30.59	33.29	36.03
DEPSL (Φ)	512	**45.86**	**30.52**	**21.89**	38.19	**44.07**	**49.56**	512	**44.83**	**29.07**	**21.66**	**30.67**	**33.56**	**36.41**
SIAMESEXML (Ψ)	300	44.81	-	21.94	37.56	43.69	49.75	300	42.16	-	21.35	29.01	32.68	36.03
NGAME (Ψ)	768	46.95	30.95	22.03	38.67	44.85	50.12	768	45.74	29.61	22.07	30.38	33.89	36.95
UNIDEC ($\Phi \oplus \Psi$)	768	**47.80**	**32.29**	**23.35**	**40.28**	**47.03**	**53.24**	768	**47.69**	**30.74**	**22.81**	**35.45**	**38.02**	**40.71**
Short-text →		LF-WikiTitles-500K							LF-WikiSeeAlsoTitles-320K					
GRAPHSAGE (Φ)	768	27.30	17.17	12.96	21.56	21.84	23.50	768	27.19	15.66	11.30	22.35	19.31	19.15
NGAME (Φ)	768	29.68	18.06	12.51	23.18	22.08	21.18	768	30.79	20.34	15.36	**25.14**	**26.77**	**28.73**
DEPSL (Φ)	512	**49.66**	**27.93**	**19.62**	**27.44**	**25.64**	**24.94**	512	**33.91**	**21.92**	**16.48**	24.22	25.80	27.99
NGAME (Ψ)	768	39.04	23.10	16.08	23.12	23.31	23.03	768	32.64	22.00	16.60	24.41	27.37	29.87
CASCADEXML (Ψ)	768	47.29	26.77	19.00	19.19	19.47	19.75	768	23.39	15.71	12.06	12.68	15.37	17.63
UNIDEC ($\Phi \oplus \Psi$)	768	**50.22**	**28.76**	**20.32**	**25.90**	**25.20**	**24.85**	768	**36.28**	**23.23**	**17.31**	**26.31**	**27.81**	**29.90**
Short-text →		LF-AmazonTitles-131K							LF-AmazonTitles-1.3M					
DPR(Φ)	768	41.85	28.71	20.88	38.17	43.93	49.45	768	44.64	39.05	34.83	32.62	35.37	36.72
ANCE (Φ)	768	42.67	29.05	20.98	38.16	43.78	49.03	768	46.44	41.48	37.59	31.91	35.31	**37.25**
NGAME (Φ)	768	**42.61**	28.86	20.69	**38.27**	**43.75**	**48.71**	768	45.82	39.94	35.48	**33.03**	**35.63**	36.80
DEPSL (Φ)	512	42.34	**28.98**	**20.87**	37.61	43.01	47.93	384	**54.20**	**48.20**	**43.38**	30.17	34.11	36.25
SIAMESEXML (Ψ)	300	41.42	30.19	21.21	35.80	40.96	46.19	300	49.02	42.72	38.52	27.12	30.43	32.52
NGAME (Ψ)	768	**44.95**	**29.87**	**21.20**	38.25	43.75	48.42	768	54.69	47.76	42.80	28.23	32.26	34.48
UNIDEC ($\Phi \oplus \Psi$)	768	44.35	29.49	21.03	**39.23**	**44.13**	**48.90**	512	**57.41**	**50.75**	**45.89**	**30.10**	**34.32**	**36.78**

Table 2.2: Experimental results showing the effectiveness of **Depsl** and **UniDEC** against both state-of-the-art dual encoder approaches and extreme classifiers. The best-performing results are put in **bold**. DE and classifier results are compared separately.

observed as the impact of explicitly mining hard-negative labels per instance instead of solely relying on the random in-batch negatives. Even though, these approaches use 12L-BERT-BASE instead of 6L-DISTILBERT common in XMC methods, ANCE only shows marginal gains over NGAME on both datasets. Our proposed DE method, DEPSL, despite using half the # Layers and half the search embedding dimension, is able to surpass these DR approaches by $15-20\%$ for P@K metrics over LF-AmazonTitles-1.3M dataset.

Search Dimensionality As mentioned before, DEPSL outperforms NGAME on P@K metrics across benchmarks. Notably, DEPSL does so by projecting (using $g_1(\cdot)$) and training the encoder embeddings in a low-dimension space of $d = 384$. Similarly, for UNIDEC, inference is carried out by concatenating $\mathfrak{N}(\Phi_{\mathfrak{C}})$ and $\Phi_{\mathfrak{D}}$ embeddings. Here, both $g_1(\cdot)$ and $g_2(\cdot)$ consist of linear layers projecting $\Phi(\cdot)$ into a low-dimensional space of $d = 256$ or $d = 384$. On the other hand, all aforementioned baselines use a higher dimension of 768 for both DE and CLF evaluations. For the proprietary Query2Bid-450M dataset, we use final dimension of 64 for all the methods necessitated by constraints of online serving.

2.4.2 Efficiency Comparison with Spectrum of XMC methods

In this section, we provide a comprehensive comparison (refer Table 2.3) of our proposed DePSL and UniDEC with two extreme ends of XMC spectrum (refer Figure 2.1): (i) Renee, which is initialized with pre-trained Ngame encoder, trains OvA classifiers with the BCE loss and, (ii) DEXML which achieves SOTA performance by training a DE using their proposed loss function *decoupled softmax*. Note that, these approaches do not pose a fair comparison with our proposed approaches as both Renee and Dexml do not use a label shortlist and backpropagate over the entire label space, requiring an order of magnitude higher GPU VRAM to run an iteration on LF-AmazonTitles-1.3M. Therefore, for the same encoder, they can be considered as the upper bound of empirical performance of CLF (OvA) and DE methods respectively. Table 2.3 shows that similar, and perhaps better, performance is possible by using our proposed UniDEC and leveraging the proposed *PSL* reduction of multi-class losses over a label shortlist.

Comparison with Renee : We observe that UniDEC delivers matching performance over P@K and PSP@K metrics on long-text datasets and significantly outperforms Renee on LF-AmazonTitles-1.3M. In fact, our proposed DE method outperforms Renee on LF-Wikipedia-500K without even employing classifiers. We posit that UniDEC is therefore more effective for skewed datasets, with higher avg. points per label and more tail labels. Furthermore, these observations imply while Renee helps BCE loss reach it's empirical limits by scaling over the entire label space, with the UniDEC framework, we can match this limit with a shortlist that is $86-212\times$ smaller than the label space, thereby consuming significantly lower compute ($1 \times$ A6000 vs $8 \times$ V100).

DePSL vs Dexml (with shortlist): While DePSL leverages the proposed *PSL* reduction in the UniDEC framework, the latter uses the POL reduction with the same loss function. As evident in the LF-AmazonTitles-1.3M, Table 2.3, (i) For a comparable label pool size (4000 vs 8192), DePSL significantly outperforms DEXML by $\sim$20% in P@K metrics. (ii) To achieve similar performance as DePSL, DEXML need to use an

Method	P@1	P@5	PSP@1	PSP@5	$\|Q_{\mathcal{B}}\|$	$\|L_{\mathcal{B}}\|$	VRAM	TT
w Classifiers	**LF-Amazon-131K**							
RENEE	**48.05**	23.26	39.32	**53.51**	512	131K	128	58
UNIDEC	47.80	**23.35**	**40.28**	53.24	576	**3000**	**48**	24
w Classifiers	**LF-WikiSeeAlso-320K**							
RENEE	**47.70**	**23.82**	31.13	40.37	2048	320K	128	81
UNIDEC	47.69	22.81	**35.45**	**40.71**	677	**3500**	**48**	39
	LF-Wikipedia-500K							
DEXML	77.71	43.32	-	-	2048	2048	80	-
DEXML	84.77	**50.31**	-	-	2048	22528	160	-
DEPSL	**85.20**	49.88	**45.96**	**59.31**	221	**3000**	**48**	55
RENEE	84.95	51.68	39.89	56.70	2048	500K	320	39
DEXML-FULL	85.78	50.53	46.27	58.97	2048	500K	320	39
Dual Encoder	**LF-AmazonTitles-1.3M**							
DEXML	42.15	32.97	-	-	8192	8192	160	-
DEXML	54.01	42.08	28.64	33.58	8192	90112	320	-
DEPSL	**54.20**	**43.38**	**30.17**	**36.25**	2200	**4000**	**48**	53
RENEE	56.04	45.32	28.56	36.14	1024	1.3M	256	105
UNIDEC	**57.41**	**45.89**	**30.10**	**36.78**	1098	**3000**	**48**	78
DEXML-FULL	58.40	45.46	31.36	36.58	8192	1.3M	640	66

Table 2.3: Experimental results showing the effectiveness of DEPSL and UNIDEC against the two ends of XMC spectrum. $|Q_{\mathcal{B}}|$ denotes batch size, $|L_{\mathcal{B}}|$ denotes label pool size and TT denotes Training Time(in hrs). Note, these comparisons are not fair owing to the significant gap in used resources.

effective label pool size of 90K. However in the same setting, DEPSL needs only $1/4^{th}$ batch size and $1/22^{th}$ label pool size. (iii) Moreover, even after scaling to an effective label pool size of 90×10^3 i.e about $20\times$ larger than DEPSL, DEXML still lacks by $1-1.5\%$ on PSP@K metrics despite having better P@K metrics. We defer the discussion about this trade-off to textbf **P vs PSP trade-off**. A similar trend is seen in LF-Wikipedia-500K. These observations empirically demonstrate the informativeness of the batches in UNIDEC - the same information can be captured by it with significantly smaller batch sizes.

UniDEC vs DEXML-Full: UNIDEC, scales to LF-AmazonTitles-1.3M on a single A6000 GPU using a label shortlist of only 3000 labels, as opposed to DEXML-FULL which requires 16 A100s and uses the entire label space of 1.3M. Despite this, Table 2.3 indicates

that UniDEC matches DEXML-Full performance on P@5 and PSP@5 metrics.

	LF-AmazonTitles-1.3M				LF-WikiTitles-500K			
Method	**P@1**	**P@5**	**PSP@1**	**PSP@5**	**P@1**	**P@5**	**PSP@1**	**PSP@5**
	DE loss - SupCon; CLF loss - SupCon							
UniDEC	53.41	43.57	32.54	38.20	48.38	19.89	26.26	24.78
UniDEC-de	49.35	39.23	27.78	32.86	48.63	19.30	27.21	24.52
UniDEC-clf	46.90	38.62	31.23	35.28	29.48	14.21	18.97	18.82
	DE loss - SupCon; CLF loss - SupCon + BCE							
UniDEC	54.86	44.61	28.05	35.05	49.68	20.15	25.29	24.67
UniDEC-de	51.08	40.78	28.64	34.00	47.07	18.88	27.47	24.53
UniDEC-clf	53.48	42.76	26.24	32.81	45.07	17.96	19.01	19.68
	DE loss - Decoupled Softmax; CLF loss - BCE							
UniDEC	55.12	44.80	31.72	37.28	48.65	19.58	26.15	24.37
UniDEC-de	50.83	40.61	27.14	33.66	47.70	18.59	26.84	24.19
UniDEC-clf	54.69	42.81	30.74	36.48	43.27	16.55	19.41	18.29
	DE loss - Decoupled Softmax; CLF loss - Decoupled Softmax							
UniDEC	56.73	45.19	**34.03**	**39.54**	48.97	19.82	**27.08**	**24.89**
UniDEC-de	52.52	42.02	29.78	35.06	49.20	19.30	27.36	24.49
UniDEC-clf	43.67	36.85	32.68	37.07	30.27	13.74	18.95	17.75
	DE loss - Decoupled Softmax; CLF loss - Decoupled Softmax + BCE							
UniDEC	**57.41**	**45.89**	30.10	36.78	**50.22**	**20.32**	25.90	24.85
UniDEC-de	52.51	42.00	29.82	35.08	49.16	19.33	27.35	24.54
UniDEC-clf	55.56	44.10	29.15	35.49	44.66	17.38	20.56	19.62

Table 2.4: Experimental results showing the effect of different loss functions while training UniDEC. Further, the table also shows the scores of inference done using only the DE head $\Phi_{\mathfrak{D}}(\mathbf{x})$ or the normalized CLF head $\mathfrak{N}(\Phi_{\mathfrak{C}}(\mathbf{x}))$, instead of the concatenated vector.

Evaluation with Multiple Loss Functions : As mentioned previously, any loss function can be chosen in the UniDEC framework, however, we experiment with two multi-class losses in particular, namely *SupCon* loss (SC) (Khosla et al., 2020) and *Decoupled Softmax* (DS) (Gupta et al., 2023). Replacing ℓ_{MC} with these gives

$$\mathcal{L}_{SupCon} = \sum_{i \in Q_{\mathcal{B}}} \frac{-1}{|\mathcal{P}_i^{\mathcal{B}}|} \sum_{p \in \mathcal{P}_i^{\mathcal{B}}} \log \frac{\exp(\langle \Phi_{\mathfrak{D}}(\mathbf{x}_i), \Phi_{\mathfrak{D}}(\mathbf{z}_p) \rangle / \tau)}{\sum_{l \in L_{\mathcal{B}}} \exp(\langle \Phi_{\mathfrak{D}}(\mathbf{x}_i), \Phi_{\mathfrak{D}}(\mathbf{z}_l) \rangle / \tau)}$$

$$\mathcal{L}_{DecoupledSoftmax} = \sum_{i \in Q_{\mathcal{B}}} \frac{-1}{|\mathcal{P}_i^{\mathcal{B}}|} \sum_{p \in \mathcal{P}_i^{\mathcal{B}}} \log \frac{\exp(\langle \Phi_{\mathfrak{D}}(\mathbf{x}_i), \Phi_{\mathfrak{D}}(\mathbf{z}_p) \rangle / \tau)}{\sum_{l \in L_{\mathcal{B}} / \{\mathcal{P}_i^{\mathcal{B}} - p\}} \exp(\langle \Phi_{\mathfrak{D}}(\mathbf{x}_i), \Phi_{\mathfrak{D}}(\mathbf{z}_l) \rangle / \tau)}$$

Notably, from Table 2.4 and Table 2.6, we observe that *Decoupled Softmax* turns out

to be a better loss for XMC tasks as it helps the logits scale better (Gupta et al., 2023) as compared to *SupCon* which caps the gradient due to a hard requirement of producing a probability distribution. We further observe that classifier performance can further improve by adding BCE loss as an auxiliary OvA loss to the classifier loss. While this helps enhance P@K metrics, the PSP@K metrics take a significant dip on the inclusion of auxiliary BCE loss. These observations are in line with the performance of *Renee* which leverages BCE loss and suffers on PSP@K metrics. Simply using BCE loss for classifier works in our pipeline, however, ends up performing worse than using multi-class loss to train the classifiers.

2.4.3 Ablation Study and Discussion

Method	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
	LF-Amazon-131K						**w/o Hard Negatives**					
UniDEC	47.80	32.29	23.35	40.28	47.03	53.24	47.45	31.49	22.53	41.28	46.93	52.40
UniDEC-de	45.24	30.32	21.97	37.85	43.92	49.90	45.45	30.29	21.79	38.15	43.97	49.53
UniDEC-clf	47.83	32.31	23.32	40.56	47.17	53.21	43.77	28.30	19.90	39.80	43.64	47.78
	LF-WikiSeeAlso-320K						**w/o Hard Negatives**					
UniDEC	47.69	30.74	22.81	35.45	38.02	40.71	46.74	30.04	22.27	35.85	38.10	40.97
UniDEC-de	44.65	28.84	21.53	30.54	33.30	36.20	43.07	27.67	20.69	30.52	32.81	35.55
UniDEC-clf	42.04	25.89	18.89	33.63	34.07	35.60	41.05	25.47	18.80	33.52	34.01	35.82
	LF-WikiTitles-500K						**w/o Hard Negatives**					
UniDEC	50.22	28.76	20.32	25.90	25.20	24.85	49.84	28.31	19.95	26.41	25.44	24.99
UniDEC-de	49.16	27.51	19.33	27.35	25.24	24.54	46.87	25.60	17.83	27.38	24.64	23.81
UniDEC-clf	44.66	24.81	17.38	20.56	19.86	19.62	44.20	24.83	17.48	21.33	20.60	20.42
	LF-AmazonTitles-1.3M						**w/o Hard Negatives**					
UniDEC	57.41	50.75	45.89	30.10	34.32	36.78	56.51	49.77	44.94	31.90	35.89	38.21
UniDEC-de	52.51	46.66	42.00	29.82	33.21	35.08	49.71	43.87	39.37	30.40	33.49	35.20
UniDEC-clf	55.56	48.77	44.10	29.15	33.15	35.49	53.31	47.21	42.90	30.41	34.19	36.47

Table 2.5: Experimental results showing the effect of adding ANNS-mined hard negatives while training UniDEC. Further, the table also shows the scores of inference done using either the DE head embedding $\Phi_{\mathfrak{D}}(\mathbf{x})$ or the normalized CLF head embedding $\mathfrak{N}(\Phi_{\mathfrak{C}}(\mathbf{x}))$, instead of the concatenated vector $\{\Phi_{\mathfrak{D}}(\mathbf{x}) \oplus \mathfrak{N}(\Phi_{\mathfrak{C}}(\mathbf{x}))\}$. The P vs PSP trade-off associated with adding ANNS-mined hard-negatives is clear by observing the underlined values.

We show the effect of the two individual components $\Phi_{\mathfrak{D}}$ and $\Phi_{\mathfrak{C}}$ of UniDEC in Table 2.5. The scores are representative of the evaluation of the respective component of the UniDEC framework, (i) UniDEC-de ($\Phi_{\mathfrak{D}}$) performs inference with an ANNS built over $\Phi_{\mathfrak{D}}(\mathbf{z}_l)|_{l=0}^{L}$, (ii) UniDEC-clf ($\Phi_{\mathfrak{C}}$) performs inference with an ANNS built

over $\mathfrak{N}(\Psi(l))|_{l=0}^{L}$ and (iii) UniDEC uses the ANNS built over the concatenation of both $\{\Phi_{\mathfrak{D}}(\mathbf{z}_l) + \mathfrak{N}(\Psi(l))\}|_{l=0}^{L}$. Notably, concatenation of embeddings leads to a more effective retrieval. We attribute its performance to two aspects, (i) as seen in previous XMC models, independent classifier weights significantly improve the discriminative capabilities of these models and (ii) we hypothesise that normalized and unnormalized spaces learn complementary information which leads to enhanced performance when an ANNS is created on their aggregation. Note that, the individual search dimensions of UniDEC-DE and UniDEC-CLF are $d/2$ and searching with a concatenated embedding leads to a fair comparison with other baselines which use a dimensionality of d.

Effect of ANNS-mined Hard Negatives The effect of explicitly adding ANNS-mined hard negatives is shown via a vis-a-vis comparison with UniDEC **(w/o Hard Negatives)** in Table 2.5. Here, when we do not add hard negatives, we compensate by adding other positives of the batched queries. More broadly, we observe a P vs PSP trade-off in this ablation. We find that not including hard negatives in the shortlist performs better on PSP@K metrics, due to inclusion of more positive labels. Consequently, adding (typically $\eta = 6$) hard negatives generally increases performance on P@K metrics, while compromising on PSP@K metrics. While the smaller datasets show only marginal improvements with added hard negatives, these effects are more pronounced in the larger datasets, proving its necessity in the pipeline.

Loss	Dual	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5	P@1	P@3	P@5	PSP@1	PSP@3	PSP@5
		LF-Amazon-131K						LF-WikiSeeAlso-320K					
SC		44.41	29.84	21.63	36.89	43.00	48.90	43.79	28.31	21.08	29.47	32.15	34.89
SC	✓	45.03	30.24	21.93	37.66	43.81	49.78	44.32	28.84	21.57	30.16	33.14	36.11
DS		45.86	30.52	21.89	38.19	44.07	49.56	44.73	28.78	21.43	30.18	32.79	35.58
DS	✓	**45.79**	**30.60**	**21.95**	**38.43**	**44.42**	**49.92**	**44.83**	**29.07**	**21.66**	**30.67**	**33.56**	**36.41**
		LF-AmazonTitles-1.3M						LF-WikiTitles-500K					
SC		52.22	46.45	41.80	29.15	32.90	34.91	48.30	27.33	19.26	27.00	25.12	24.51
SC	✓	50.62	45.09	40.64	30.51	34.30	36.33	47.33	26.79	18.94	27.41	25.21	24.63
DS		**54.20**	**48.20**	**43.38**	30.17	34.11	36.25	**49.66**	**27.93**	**19.62**	27.44	25.64	24.94
DS	✓	53.26	47.55	42.86	**31.90**	**35.80**	**37.88**	48.87	27.47	19.35	**28.09**	**25.77**	**25.08**

Table 2.6: Experimental results showing the effect of adding symmetric loss while training DePSL.

Effect of Symmetric Loss As shown in Table 2.6, making the loss function symmetric has a favorable effect on all metrics for *long-text* datasets. However, this make the *short-text* datasets favor PSP@K metrics more, at an expense of P@K metrics. We believe this happens because of mixing data distributions. While adding a *short-text* loss over *long-text* document helps the model understand the label distribution better, this has a reverse effect on *short-text* datasets and the label distribution confuses with already *short-text* query distribution and ends up learning the label distribution more at the expense of query distribution.

2.4.4 Query2Bid Evaluation and Live A/B testing on Sponsored Search

To demonstrate the effectiveness of our method on proprietary datasets and real-world scenarios, we do experiments in sponsored search setting. The proposed model was evaluated on proprietary dataset for matching queries to advertiser bid phrases (Query2Bid) consisting of 450M labels. Query2Bid-450M dataset was created by mining the logs from search engine and enhancing it through Data Augmentation techniques using a ensemble of leading (proprietary) algorithms such as Information Retrieval models (IR), Dense retrieval models (DR), Generative Non-Autoregressive models (NAR), Extreme-Multi-label Classification models (XMC) and even GPT Inference techniques.

Experimental Setup : The BERT Encoder is initialized with 6-Layer DistilBERT base architecture. Since the search queries and bid phrases are of short-text in nature, a max-sequence-length of 12 is used. We evaluate DEPSL against XMC and DR models deployed in production which could scale to the magnitude of chosen dataset. Training batch-size is set to 2048 and other Hyperparameters are chosen to be same as for public benchmark datasets. Training is carried out on **8 V100 GPUs** and could easily complete within **48 hours**. Performance is measured using popular metrics such as Precision@K (P@K) with $K \in 1, 3, 5, 10$.

Method	P@1	P@3	P@5	P@10
NGAME	86.16	73.07	64.61	51.94
SimCSE	86.08	73.26	65.27	53.51
DePSL	**87.33**	**74.63**	**66.44**	**54.13**

Table 2.7: Results on Query2Bid-450M dataset for Sponsored Search

Offline Results : Table 2.7 shows that on DePSL can be 1.15-1.83% more accurate than the leading DR & XMC methods in Sponsored Search setting. This indicates that leveraging DePSL can yield superior gains in real-world search applications.

Live A/B Testing in a Search Engine: DePSL was deployed on Live Search Engine and A/B tests were performed on real-world traffic. The effect of adding DePSL to the ensemble of existing models in the system was measured through popular metrics such as Impression Yield (IY), Click Yield (CY), Click-Through Rate (CTR) and Query Coverage (QC). Refer (Dahiya et al., 2023a) for definitions and details about these metrics. DePSL was observed to improve IY, CY, CTR and QC by **0.87%, 0.66%, 0.21% and 1.11%** respectively. Gains in IY, CY and CTR establish that DePSL is able to predict previously unmatched relations and the predictions are more relevant to the end user. QC boost indicates that DePSL is able to serve matches for queries to which there were no matches before in the system. This ascertains the zero-shot capabilities of the model.

2.5 Other Related Works

To reduce computational costs of training classifiers, previous XMC methods tend to make use of various shortlisting strategies, which serves as a *good approximation* to the loss over the entire label space (Chang et al., 2020; Dahiya et al., 2021b; You et al., 2019). This shortlist can be created in one of the two ways : (i) by training a meta classifier on coarser levels of a hierarchically-split probabilistic label tree. The leaf nodes of the top-k nodes constitute the shortlist (Jiang et al., 2021; Kharbanda et al., 2023, 2022) (ii) by retrieving the top-k labels for a query from an ANNS built on the label representations

from a contrastively trained DE (Dahiya et al., 2021a). Both these methods have different trade-offs. The meta-classifier based approach has a higher memory footprint due to the presence of additional meta classifier ($\sim \mathbb{R}^{L/10 \times d}$ in size) along with the extreme classifier, but it gives enhanced performance since this provides progressively harder negatives in a dynamic shortlist, varying every epoch (Jiang et al., 2021; Kharbanda et al., 2023, 2022). The shortlisting based on ANNS requires training the model in multiple stages, which has low memory usage, but needs longer training schedules and uses a static shortlist for training extreme classifiers (Dahiya et al., 2023a, 2021b; Mittal et al., 2021a,b).

Previous research has also explored various other methods : (i) label trees (Khandagale et al., 2020; Prabhu et al., 2018a; Wydmuch et al., 2018), (ii) classifiers based on hierarchical label trees (Chang et al., 2020; Zhang et al., 2021b; Prabhu et al., 2018b). Alongside previous negative-mining works, the statistical consequences of this sampling (Reddi et al., 2018) and missing labels (Jain et al., 2016; Qaraei et al., 2021; Schultheis et al., 2022; Schultheis and Babbar, 2021; Wydmuch et al., 2021) have led to novel insights in designing unbiased loss functions - which can also be applied in UniDEC.

2.6 Conclusion

In this paper, we present a new loss-independent end-to-end XMC framework, UniDEC, that aims to leverage the best of both, a dual encoder and a classifier in a compute-efficient manner. The dual-encoder is used to mine hard negatives, which are in turn used as the shortlist for the classifier, eliminating the need for meta classifiers. Highly informative in-batch labels are created which maximise the supervisory signals while keeping the GPU memory footprint as low as possible - to the extent that we outperform previous SOTAs with just a single GPU. The dual encoders and classifiers are unified and trained with the same multi-class loss function, which follows the proposed *pick-some-labels* paradigm. To the best of our knowledge, we are the first work to study the effect of PAL-like losses for training XMC classifiers. We hope this inspires future works to study the proposed *PSL* reduction for multilabel problems as a compute-efficient means to further eliminate the

need of high-capacity classifiers in XMC, bringing the scope of this problem closer to the more general dense retrieval regime.

Adamic, L. A. and Huberman, B. A. (2002). Zipf's law and the internet. *Glottometrics*, 3(1):143–150. 1, 24

Babbar, R. and Schölkopf, B. (2017). DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. In *WSDM*. 21, 24

Babbar, R. and Schölkopf, B. (2019). Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108:1329–1351. 11

Bhatia, K., Dahiya, K., Jain, H., Kar, P., Mittal, A., Prabhu, Y., and Varma, M. (2016). The extreme classification repository: Multi-label datasets and code. 35

Buvanesh, A., Chand, R., Prakash, J., Paliwal, B., Dhawan, M., Madan, N., Hada, D., Jain, V., MEHTA, S., Prabhu, Y., Gupta, M., Ramjee, R., and Varma, M. (2024). Enhancing tail performance in extreme classifiers by label variance reduction. In *The Twelfth International Conference on Learning Representations*. 3, 11

Chang, W.-C., Yu, H.-F., Zhong, K., Yang, Y., and Dhillon, I. (2020). Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *KDD*. 25, 29, 44, 45

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 257–266, New York, NY, USA. Association for Computing Machinery. 1

Chien, E., Zhang, J., Hsieh, C.-J., Jiang, J.-Y., Chang, W.-C., Milenkovic, O., and Yu, H.-F. (2023). Pina: Leveraging side information in extreme multi-label classification via predicted instance neighborhood aggregation. *arXiv preprint arXiv:2305.12349*. 3, 21

Dahiya, K., Agarwal, A., Saini, D., K, G., Jiao, J., Singh, A., Agarwal, S., Kar, P., and Varma, M. (2021a). Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2330–2340. PMLR. 2, 3, 4, 5, 7, 9, 12, 21, 22, 26, 28, 29, 30, 45

Dahiya, K., Gupta, N., Saini, D., Soni, A., Wang, Y., Dave, K., Jiao, J., K, G., Dey, P., Singh, A., et al. (2023a). Ngame: Negative mining-aware mini-batching for extreme classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 258–266. 2, 3, 4, 5, 7, 12, 21, 22, 25, 26, 28, 29, 31, 35, 44, 45

Dahiya, K., Saini, D., Mittal, A., Shaw, A., Dave, K., Soni, A., Jain, H., Agarwal, S., and Varma, M. (2021b). Deepxml: A deep extreme multi-label learning framework applied to short text documents. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 31–39, New York, NY, USA. Association for Computing Machinery. 1, 2, 4, 12, 25, 30, 33, 44, 45

Dahiya, K., Yadav, S., Sondhi, S., Saini, D., Mehta, S., Jiao, J., Agarwal, S., Kar, P., and Varma, M. (2023b). Deep encoders with auxiliary parameters for extreme classification. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 358–367. 25, 29

Dembczyński, K., Waegeman, W., Cheng, W., and Hüllermeier, E. (2012). On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88:5–45. 7, 8

Gunel, B., Du, J., Conneau, A., and Stoyanov, V. (2021). Supervised contrastive learning for pre-trained language model fine-tuning. 34

Guo, C., Mousavi, A., Wu, X., Holtmann-Rice, D. N., Kale, S., Reddi, S., and Kumar, S. (2019). Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces. In *NeurIPS*. 3, 5, 11

Gupta, N., Khatri, D., Rawat, A. S., Bhojanapalli, S., Jain, P., and Dhillon, I. S. (2023). Efficacy of dual-encoders for extreme multi-label classification. 22, 26, 28, 35, 40, 41

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc. 1

Hüllermeier, E., Wever, M., Loza Mencia, E., Fürnkranz, J., and Rapp, M. (2022). A flexible class of dependence-aware multi-label loss functions. *Machine Learning*, 111(2):713–737. 8

Jain, H., Balasubramanian, V., Chunduri, B., and Varma, M. (2019). Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 1

Jain, H., Prabhu, Y., and Varma, M. (2016). Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *KDD*, pages 935–944. 13, 16, 45

Jain, V., Prakash, J., Saini, D., Jiao, J., Ramjee, R., and Varma, M. (2023). Renee: End-to-end training of extreme classification models. *Proceedings of Machine Learning and Systems*. 12, 22, 25, 35

Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z., and Zhuang, F. (2021). Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label

text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7987–7994. 25, 27, 44, 45

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics. 22

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020a). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*. 22

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020b). Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics. 28, 35

Khandagale, S., Xiao, H., and Babbar, R. (2020). Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109(11):2099–2119. 21, 45

Kharbanda, S., Banerjee, A., Gupta, D., Palrecha, A., and Babbar, R. (2023). Inceptionxml: A lightweight framework with synchronized negative sampling for short text extreme classification. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 760–769, Taipei, Taiwan. Association for Computing Machinery. 2, 4, 12, 14, 25, 27, 44, 45

Kharbanda, S., Banerjee, A., Schultheis, E., and Babbar, R. (2022). Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. In *Advances in Neural Information Processing Systems*, volume 35, pages 2074–2087. Curran Associates, Inc. 21, 25, 27, 44, 45

Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48. 22

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673. 34, 40

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466. 22

Lu, Y., Liu, Y., Liu, J., Shi, Y., Huang, Z., Sun, S. F. Y., Tian, H., Wu, H., Wang, S., Yin, D., et al. (2022). Ernie-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval. *arXiv preprint arXiv:2205.09153*. 22

Menon, A. K., Rawat, A. S., Reddi, S., and Kumar, S. (2019). Multilabel reductions: what is my loss optimising? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. 8, 26, 30

Mittal, A., Dahiya, K., Agrawal, S., Saini, D., Agarwal, S., Kar, P., and Varma, M. (2021a). Decaf: Deep extreme classification with label features. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 49–57, New York, NY, USA. Association for Computing Machinery. 1, 2, 3, 5, 12, 21, 45

Mittal, A., Sachdeva, N., Agrawal, S., Agarwal, S., Kar, P., and Varma, M. (2021b). Eclare: Extreme classification with label graph correlations. In *Proceedings of the Web Conference 2021*, WWW '21, page 3721–3732, New York, NY, USA. Association for Computing Machinery. 2, 3, 5, 12, 21, 45

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). Ms marco: A human-generated machine reading comprehension dataset. 22

Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., Androutsopoulos, I., Amini, M.-R., and Galinari, P. (2015). Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*. 1

Prabhu, Y., Kag, A., Gopinath, S., Dahiya, K., Harsola, S., Agrawal, R., and Varma, M. (2018a). Extreme multi-label learning with label features for warm-start tagging, ranking and recommendation. In *WSDM*. 45

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., and Varma, M. (2018b). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 993–1002, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. 1, 45

Qaraei, M., Schultheis, E., Gupta, P., and Babbar, R. (2021). Convex surrogates for unbiased loss functions in extreme classification with missing labels. In *Proceedings of the Web Conference 2021*, pages 3711–3720. 13, 45

Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W. X., Dong, D., Wu, H., and Wang, H. (2021). RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics. 22, 28, 35

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. 32

Reddi, S. J., Kale, S., Yu, F., Rice, D. N. H., Chen, J., and Kumar, S. (2018). Stochastic Negative Mining for Learning with Large Output Spaces. *CoRR*. 45

Ren, R., Qu, Y., Liu, J., Zhao, W. X., She, Q., Wu, H., Wang, H., and Wen, J.-R. (2021). RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 22

Saini, D., Jain, A. K., Dave, K., Jiao, J., Singh, A., Zhang, R., and Varma, M. (2021). Galaxc: Graph neural networks with labelwise attention for extreme classification. In *ACM International World Wide Web Conference*. 21

Schultheis, E. and Babbar, R. (2021). Unbiased loss functions for multilabel classification with missing labels. *arXiv preprint arXiv:2109.11282*. 45

Schultheis, E. and Babbar, R. (2022). Speeding-up one-versus-all training for extreme classification via mean-separating initialization. *Machine Learning*, 111(11):3953–3976. 21

Schultheis, E., Wydmuch, M., Babbar, R., and Dembczynski, K. (2022). On missing labels, long-tails and propensities in extreme multi-label classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1547–1557. 13, 24, 45

Schultheis, E., Wydmuch, M., Kotlowski, W., Babbar, R., and Dembczynski, K. (2024). Generalized test utilities for long-tail performance in extreme multi-label classification. *Advances in Neural Information Processing Systems*, 36. 13

Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M. R., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., et al. (2015). An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1–28. 1

Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., and Dembczynski, K. (2018). A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NIPS*. 45

Wydmuch, M., Jasinska-Kobus, K., Babbar, R., and Dembczynski, K. (2021). Propensity-scored probabilistic label trees. In *SIGIR*, pages 2252–2256. 45

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. (2020). Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*. 22, 35

Ye, H., Chen, Z., Wang, D.-H., and D., D. B. (2020). Pretrained Generalized Autoregressive Model with Adaptive Probabilistic Label Clusters for Extreme Multi-label Text Classification. In *ICML*. 1

You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H., and Zhu, S. (2019). Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *NeurIPS*. 3, 21, 29, 44

Zhang, H., Gong, Y., Shen, Y., Lv, J., Duan, N., and Chen, W. (2021a). Adversarial retriever-ranker for dense text retrieval. *arXiv preprint arXiv:2110.03611*. 22

Zhang, J., Chang, W.-C., Yu, H.-F., and Dhillon, I. (2021b). Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34:7267–7280. 29, 45

Zhang, J., Chang, W.-C., Yu, H.-F., and Dhillon, I. S. (2021c). Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In *Advances in Neural Information Processing Systems*. 3, 4, 21