

Addressing statistical and computational challenges in extreme multilabel classification

with unbiased estimators, macro-averaged metrics, and hardware-aware implementations

Erik Schultheis



Aalto University publication series
Doctoral Theses 180/2025

Addressing statistical and computational challenges in extreme multilabel classification

with unbiased estimators, macro-averaged metrics, and hardware-aware implementations

Erik Schultheis

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall AS2 (Maarintie 8, Espoo) of the school on the 2nd of October 2025 at 15:00.

Aalto University
School of Science
Department of Computer Science

Supervising professor

Professor Pekka Marttinen, Aalto University, Finland

Thesis advisors

Professor Rohit Babbar, Aalto University, Finland and University of Bath, UK

Preliminary examiners

Dr. Aditya Krishna Menon, Google, USA

Prof. Inderjit Dhillon, University of Texas at Austin, USA

Opponent

Dr. Aditya Krishna Menon, Google, USA

Aalto University publication series
Doctoral Theses 180/2025

© Erik Schultheis

ISBN 978-952-64-2731-7 (soft cover)

ISBN 978-952-64-2730-0 (PDF)

ISSN 1799-4934 (printed)

ISSN 1799-4942 (PDF)

<https://urn.fi/URN:ISBN:978-952-64-2730-0>

Unigrafia Oy
Helsinki 2025

Author Erik Schultheis

Name of the doctoral thesis Addressing statistical and computational challenges in extreme multilabel classification with unbiased estimators, macro-averaged metrics, and hardware-aware implementations

Article-based thesis

Number of pages 298

Keywords multilabel classification, missing labels, classification with large output spaces, long-tailed prediction, sparse neural networks

This thesis tackles statistical and computational challenges in extreme multilabel classification (XMC) problems, that is, in tasks where the label space is gigantic, possibly in the millions of labels. Such problems are plagued by missing labels and data scarcity, particularly in the form of tail labels, and the enormous label space turns operations that are cheap in typical machine learning problems, such as calculating the loss in the classification layer, into computationally challenging tasks.

Towards addressing the missing-label problem, this thesis derives unbiased estimators for generic multilabel loss functions under the assumption that a propensity model is available. A critical look at the propensity model that is in widespread usage in the current XMC literature is provided, in particular regarding the the problematic double role of using propensities both to compensate for missing labels and as a measure for performance on infrequent tail labels. As an alternative, macro-averaged performance metrics are proposed, and prediction algorithms aiming to optimize these metrics in two different inference frameworks are presented.

The thesis presents a new approach to train linear extreme classifiers, still an important baseline, significantly faster than before, owing to a new weight initialization scheme, and code that is aware of the memory layout of modern NUMA processors. Additionally, it presents a novel way to exploit weight sparsity, already at the training stage, to reduce the on-device memory consumption. This is achieved by combining dynamic sparse training algorithms with an efficient weight storage format that at the same time allows for a fast implementation of matrix multiplication.

Preface

This document represents the culmination of about five years of fun and exiting work as a doctoral student at Aalto University. It certainly got off to an interesting start: Move abroad, get a flat, trip to IKEA, lockdown... Still, I think we managed to make the best out of it!

Maybe it was a consequence of Covid, maybe it would have happened like this anyway, but a lot of the collaborations for the papers that comprise this thesis have been organized remotely, but in the end, I did manage to meet all of you in person. Special thanks to Krzysztof and Wojciech for inviting me to visit PUT in Poznań, and to Yani and Mike for hosting me in Calgary. Unfortunately, I missed Marek by a day or two when visiting Poland, but we finally got together during ICLR in Vienna, three years after we started working together. It's been a pleasure meeting you all.

Of course, besides external collaborators, I've had the joy of working with several people at Aalto. First and foremost, my initial supervisor and now thesis advisor Rohit Babbar. He was there when needed, offering advice and guidance, but also let me — encouraged me — to explore things that were a bit outside (at least initially) of the original project plan. The two dynamic sparse training papers presented here are an offshoot of just such experimentation, so I think it paid off. At about the midpoint of my doctorate, Rohit left Aalto for Bath university. While that meant an official demotion from thesis supervisor to thesis advisor, in practice this did not change his engagement with me or my colleagues. Still, a new supervisor was needed, and Pekka Marttinen took up the task; not just for me, but for Rohit's other three PhD students as well. When that decision was made, it was certainly a bit of a weird feeling at first, as I had not interacted much with Pekka before, but he turned out to be a great supervisor.

I'd be remiss not to mention Prof. Inderjit Dhillon and Dr. Aditya Krishna Menon, who graciously agreed to serve as pre-examiners for the thesis and provided valuable feedback on the preliminary version of this work.

My thanks also goes to my fellow doctoral students in the XMC group, Mohammadreza, Jinbin, and Nasib. We've had many interesting discussions over the course of the last couple of years. Particularly memorable were

those we had while barbecuing by the sea.

During my time at Aalto, I also took part in teaching two courses. The Programming Parallel Computers course was awesome both from the perspective of a student and as a TA/developer, many thanks to Henrik and Jukka. The Geometric Deep Learning course, though not related to any of the research presented here, was a nice opportunity to reconnect a bit with my physics background, and preparing the introductory lectures certainly was an interesting challenge. A big thank you to Alison, Çağlar, Kate, and Vikas for co-organizing the course. It couldn't have happened without you!

I'd like to mention the Aalto ScienceIT team, too. The fact that, if there is an IT problem of some form, you'll generally get an initial response, if not a full fix, within a few hours on any day of the week is amazing. Scientific computing is one thing that is taken for granted when it works well, but a source of endless frustration when it does not, and I'm certainly glad I didn't have to experience the latter.

Finally, I'd like to thank my family, my parents, brothers and sisters, as well as my friends, for all the times they were there for me during these years.

Helsinki, September 17, 2025,

Erik Schultheis

Contents

Preface	5
Contents	7
List of Publications	9
Author's Contribution	11
List of Figures	13
List of Tables	15
Abbreviations	17
Symbols	19
1. Introduction	21
1.1 Defining Extreme Multilabel Classification	21
1.2 Application Areas	23
1.3 Current Challenges	24
1.4 Contributions	27
1.5 Structure of the Thesis	28
2. Preliminaries	31
2.1 Task Losses and Multilabel Reductions	31
2.2 XMC methods	35
3. Missing Labels	41
3.1 Background: Noisy- and Missing-Label Learning	41
3.2 Unbiased estimates and their surrogates for One-vs-All loss functions	44
3.3 Unbiased estimates and their surrogates for general multi- label reductions	46
3.4 Critical Discussion of the Propensity Model	50
3.5 Summary and Discussion	53

4. Macro-averaged Performance Metrics	55
4.1 Background: Generalized performance metrics	56
4.2 Macro-averaged performance metrics	57
4.3 Optimal predictions under the ETU setting	59
4.4 Optimal predictions under the PU setting	61
4.5 Results and discussion	62
5. Computational Efficiency	65
5.1 Efficient linear models for imbalanced multilabel problems	65
5.2 Background: Memory efficiency and sparsity	67
5.3 DST for learning the classification layer	69
5.4 End-to-end DST training	71
5.5 Discussion and Outlook	73
6. Summary and Conclusion	75
References	77
Publications	93

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

- I** Mohammadreza Qaraei, Erik Schultheis, Priyanshu Gupta, and Rohit Babbar. Convex surrogates for unbiased loss functions in extreme classification with missing labels. In *WWW '21: Proceedings of the Web Conference 2021*, Ljubljana, pages 3711–3720, April 2021.
- II** Erik Schultheis and Rohit Babbar. Unbiased Loss Functions for Multilabel Classification with Missing Labels. *Accepted for publication in Transactions on Machine Learning Research*, September 2025.
- III** Erik Schultheis, Rohit Babbar, Marek Wydmuch, Krzysztof Dembczynski. 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*, 1547–1557, August 2022.
- IV** Erik Schultheis, Rohit Babbar. Speeding-up one-versus-all training for extreme classification via mean-separating initialization. *Machine Learning*, volume 111, issue 11, pp 3953–3976, November 2022.
- V** Erik Schultheis, Rohit Babbar. Towards Memory-Efficient Training for Extremely Large Output Spaces—Learning with 670k Labels on a Single Commodity GPU. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 689–704, September 2023.
- VI** Nasib Ullah, Erik Schultheis, Mike Lasby, Yani Ioannou, Rohit Babbar. Navigating Extremes: Dynamic Sparsity in Large Output Spaces. In *Advances in Neural Information Processing Systems*, Vol. 37, 2024.

VII Erik Schultheis, Marek Wydmuch, Wojciech Kotłowski, Rohit Babbar, Krzysztof Dembczynski. Generalized test utilities for long-tail performance in extreme multi-label classification. In *Advances in Neural Information Processing Systems*, Vol. 36, 2023.

VIII Erik Schultheis, Wojciech Kotłowski, Marek Wydmuch, Rohit Babbar, Strom Borman, Krzysztof Dembczyński. Consistent algorithms for multi-label classification with macro-at- k metrics. In *The Twelfth International Conference on Learning Representations*, May 2024.

Author's Contribution

Publication I: “Convex surrogates for unbiased loss functions in extreme classification with missing labels”

MQ is the main author. ES is responsible for the theoretical contributions of the paper, i.e., Section 2. MQ and PG conducted the experiments. RB helped writing and revising the manuscript.

Publication II: “Unbiased Loss Functions for Multilabel Classification with Missing Labels”

ES is the main author, responsible for the theorems and proofs as well as most of the writing of the paper. Further revisions were made jointly by ES and RB.

Publication III: “On Missing Labels, Long-tails and Propensities in Extreme Multi-label Classification”

ES provided the idea for the paper based on counterintuitive results with the existing propensity model. ES performed the analysis of shortcomings of the existing model, MW performed the experiments using alternative propensity models and joint learning. RB and KD contributed to writing the paper.

Publication IV: “Speeding-up one-versus-all training for extreme classification via mean-separating initialization”

ES is main author of the paper, conceived of and implemented the idea. RB helped revise the manuscript and provided clarifications regarding the original Dismec implementations.

Publication V: “Towards Memory-Efficient Training for Extremely Large Output Spaces—Learning with 670k Labels on a Single Commodity GPU”

ES is the main author, responsible for the idea, implementation, and experiments. ES and RB contributed to writing the paper.

Publication VI: “Navigating Extremes: Dynamic Sparsity in Large Output Spaces”

NU is the main author of the paper. NU adapted V for end-to-end training, and conducted the experiments. ES coded improved CUDA kernels to enable faster experiments. RB and ES contributed towards the pool of ideas attempting to fix DST for end-to-end XMC training. ML and YI provided comparisons against other sparsity baselines. All authors contributed towards writing the paper.

Publication VII: “Generalized test utilities for long-tail performance in extreme multi-label classification”

ES and MW are main authors of the paper. ES developed the initial version of the BCA algorithm, MW an enhanced implementation compatible with sparse predictions and conducted the experiments. WK performed the regret analysis. RB and KD contributed to discussions and writing.

Publication VIII: “Consistent algorithms for multi-label classification with macro-at- k metrics”

ES, WK, and MW are main authors of the paper. WK developed the proofs for the optimal classifier (appendix B, C), ES wrote the proofs for convergence of Frank-Wolfe-based algorithm (appendix D), and the discussion about confusion tensor measure (appendix H). MW handled the implementation and running of the experiments. SB provided the proofs for Lemmas C4 and C5. RB and KD contributed to discussions and writing.

List of Figures

3.1	Non-commutativity of taking the unbiased estimate and forming a surrogate by a piecewise-linear function.	45
3.2	Unbiased estimate of per-example recall with artificial data	49
3.3	Normalized categorical cross-entropy with missing labels .	49
3.4	Propensity estimates on Wikipedia data	51
4.1	Illustration of Frank-Wole optimization	62
4.2	Interpolation between macro and instance-wise performance measures	63
5.1	NUMA memory of Mahti compute nodes	66
5.2	Mean-separating initialization	67
5.3	Schematic depiction of different sparse matrix formats.	70
5.4	Different sparse training techniques at various sparsity levels	72

List of Tables

1.1	Statistics of publicly available extreme classification datasets	22
3.1	Normalized and unnormalized propensity-scored precision of PfastreXML	52
4.1	Performance measures on AmazonCat-13k of a classifier trained on the full set of labels and a classifier trained with only 1k head labels.	55
4.2	Examples of generalized performance measures	56

Abbreviations

BCA Block coordinate ascent

DST Dynamic sparse training

ETU Expected test utility

FW Frank-Wolfe (algorithm)

LSH Locality-sensitive hashing

NUMA Non-uniform memory access

OVA One versus all

PAL Pick-all-labels

PLT Probabilistic label tree

PU Population utility

XMC Extreme multilabel classification

Symbols

$a, A, \mathbf{a}, \mathbf{A}, A, \mathbf{A}, \mathcal{A}$ Variation of the same symbol: Scalar a , random scalar A , vector \mathbf{a} , random vector \mathbf{A} , matrix A , random matrix \mathbf{A} , set \mathcal{A} .

$C[\hat{\mathbf{Y}}, \mathbf{Y}]$ Confusion matrix

$\hat{C}(\hat{\mathbf{y}}, \mathbf{y})$ Empirical confusion matrix

\mathcal{H} Hypothesis class

H Hessian matrix

$\ell(\hat{\mathbf{y}}, \mathbf{y})$ loss function with predictions $\hat{\mathbf{y}}$ and ground truth \mathbf{y}

$\tilde{\ell}(\hat{\mathbf{y}}, \tilde{\mathbf{y}})$ missing-label adapted loss function with predictions $\hat{\mathbf{y}}$ and observed labels $\tilde{\mathbf{y}}$

\mathbf{M} Multiplicative mask for missing labels; $\tilde{\mathbf{Y}} = \mathbf{M} \odot \mathbf{Y}$

m Number of labels

n, n_j Number of instances; number of instances where label j is positive

nDCG@ k , DCG@ k (Normalized) Discounted cumulative gain at k , see (2.9) and (2.10)

\mathbf{p} Propensity vector

\mathcal{P} Set of positive labels

P Probability distribution

P@ k Precision-at- k , see (2.7)

PSP@ k Propensity-scored precision, see (2.11)

R@ k Recall-at- k , see (2.7)

$R[\phi, P]$ population risk of classifier ϕ on distribution P

$\hat{R}[\phi, \mathcal{D}]$ empirical risk of classifier ϕ on dataset \mathcal{D}

Symbols

$\hat{\mathbf{s}}$ Prediction scores

t, T Noise transition function; noise transition matrix

\mathcal{X} Instance space

$\mathbf{x}, \mathbf{X}, \mathcal{X}, \mathbf{X}$ Input example/instance/data point

$y, Y, \mathbf{y}, \mathbf{Y}, Y, \mathbf{Y}$ Ground-truth labels

$\tilde{y}, \tilde{Y}, \tilde{\mathbf{y}}, \tilde{\mathbf{Y}}$ Observed labels

$\hat{\mathbf{y}}$ Predicted labels

$\boldsymbol{\eta}(\mathbf{x})$ Marginal label probabilities for a given input example

ρ_{\pm} Binary label noise for positive (ρ_{+}) and negative (ρ_{-}) labels

ϕ XMC model

ϕ^* Bayes classifier

Ψ Generalized performance metric

$\mathbb{1}[\cdot]$ Indicator function

1. Introduction

This thesis concerns extreme multilabel classification, that is, multilabel classification problems that have a very large label space. Below, we give a broad overview over the field, starting from defining characteristics of extreme multilabel problems. Then, several application areas are introduced, and the current challenges that are addressed in this thesis are discussed and corresponding research questions formulated.

1.1 Defining Extreme Multilabel Classification

While *Extreme Multilabel Classification (XMC)* does not have a strict formal definition [9], it generally designates multilabel classification problems where the size of the label space poses a serious challenge. From a computational perspective, what constitutes a challenging label-space changed (and changes) over time, as memory and compute capacity of typical hardware resources increase. For example, in 2012, a high-end GPU like the Quadro K5000 had 4 GB of memory, whereas ten years later the RTX 6000 has 48 GB, a twelve-fold increase in memory.¹

To be less dependent on the specifications of ever-changing current-generation hardware, one can instead consider the resource cost of the classification layer in relation to the entire model; if classification represents a sizable fraction of the total compute and memory requirements, then the problem can be considered extreme. Of course, a large amount of research effort in XMC is spent precisely on bringing down these costs, by, e.g., switching from methods that are linear in the number of labels to those that have only a logarithmic dependency. Consequently, in a successful XMC method, the classification part might no longer be dominating the cost.

As an illustration, consider a standard text classification task using Bert [32]. In its bert-base version, this model has 110 million parameters, and an embedding dimension of 768. If the number of classes is small, say 1000,

¹https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units

Dataset	#Labels	#Training	#Test	PpL	LpP
EURLex-4K	3993	15539	3809	25.73	5.31
AmazonCat-13K	13330	1186239	306782	448.57	5.04
Wikipedia-500K	501070	1813391	783743	24.75	4.77
Amazon-3M	2812281	1717899	742507	31.64	36.17

Table 1.1. Statistics of publicly available extreme classification datasets [11]. “PpL” indicates the average number of positive training examples per label, “LpP” the average number of labels per point.

then this corresponds to an additional 768 thousand parameters, a mere 0.7%. But in the extreme case, for example when classifying 501070 different Wikipedia categories [11], the classification layer takes 385 million parameters, about 3 times as much as the embedding model! Chang et al. [19, Table 1] gives a similar example showing memory consumption when using an XLNet [166] backbone.

A second characterization that is independent on the resources available to the researcher, instead identifies XMC problems based on dataset statistics, examples of which are shown in Table 1.1. In that view, a problem is “extreme” if the number of training samples is of comparable order of magnitude as the number of possible labels.

Overall, the best approach might be to just collect a list of criteria, based on these alternative definitions above, that make a problem XMC-like, without trying to delineate the field precisely:

Computational feasibility a naïve approach is infeasible (or at least very costly) on current-generation hardware

Cost distribution the computational cost to do classification is comparable to, or even outweighs, the cost of feature extraction

Data scarcity the number of training points and the number of labels are of similar order of magnitude

Data imbalance the relative frequency of labels varies over many orders of magnitude; the label-distribution is long-tailed

Data quality the cost of annotating training data is prohibitively high due to the large number of labels; the training set will contain missing labels

Next, we will illustrate these properties in example XMC applications.

1.2 Application Areas

Document Classification When annotating documents with a fine-grained set of categories, the resulting classification problem can end up with a very large output space. Older examples, such as mapping European Union legal texts onto a set of 3000 categories [101] would not be considered extreme anymore; however, note that, if the problem is to be solved by transforming the *multilabel* classification task into pairwise *binary* classification tasks,² $3000^2/2 = 4500000$ individual binary classifiers need to be generated and stored, which does require special consideration even for linear classifiers [101]. More recent works on large-scale legal document classification include Chalkidis et al. [18] and Song et al. [141].

A larger example is the classification of clinical nodes into codes as specified by the International Classification of Diseases (ICD) [118, 3, 156, 179, 13, 97, 61].³ Such data is available, for instance, in the Mimic-II [135] and Mimic-III [74] databases. With such fine-grained medical diagnosis comes a huge imbalance in the relative frequencies of different classifications. Among the over 70000 codes in the US version, ICD-10-CM, are such rare events as V95.40 “Unspecific spacecraft accident injuring occupant” [104].

Of similar scale is the task of patent classification, with a total of 79039⁴ International Patent Classification (IPC) codes [76].

For even more classes, one can consider the task of automatic association of Wikipedia articles with their corresponding categories. In the Wikipedia-500k dataset [11], there are 1699722 articles and 501070 categories. This means that in order to manually annotate the entire dataset, human annotators would have to check $1699722 \times 501070 \approx 8.52 \times 10^{11}$ combinations of article and category – something that is completely infeasible. Therefore, one has to rely on data gathered in the wild, which means annotations will be noisy. More specifically, since selected categories will be displayed on Wikipedia’s website, any false positives would be quickly corrected, whereas a missing category is generally invisible, leading to one-sided label noise in the form of missing labels [64].

Recommendations A second class of applications uses XMC for different kinds of recommendation. For example, in the Amazon-670k dataset [105, 11], the goal is to predict, for a given product, with which other products it will be frequently bought together [64]. Other recommendation tasks include predicting related searches out of a corpus of 100 million possible queries [65], or matching advertisements to search phrases [2].

²Such transformations, or *reductions*, are discussed in more detail in section 2.1

³This is a non-representative sample of recent works on ICD classification that mention typical XMC challenges such as long tails. Attempts at automatic ICD identification go back at least to the 90s [88].

⁴January 2025, <https://www.wipo.int/classifications/ipc/en/ITsupport/Version20250101/transformations/stats.html>

Many of these recommendation tasks fall under the category of *short-text XMC*, that is, the inputs to the classification task are not full documents, but rather short phrases or sentences. For example, a search query usually consists only of a few words. This means that frugal architectures [80] can be sufficient to process the input, leading to essentially all of the models compute and memory cost being located in the classification layer. Additionally, short-text tasks often allow to address the data-scarcity problem because labels can be short phrases too, creating a symmetry that can be exploited in contrastive learning [26] or through dataset augmentation [23, 81].

Next-word prediction A *multiclass* extreme classification problem naturally arises in unsupervised language modelling, when models are trained to predict the next token (e.g., a word or character) in the sequence [89].

In this case, using a larger vocabulary of possible tokens means that longer sequences of text can be represented within a fixed-length token sequence. This is important as it decreases the computational cost, especially with dense attention layers that scale quadratically with the sequence length [154]. Consequently, several large language models have been trained with vocabularies exceeding 100000 tokens, e.g., LLama3 [36] achieves 3.94 characters per token with 128k vocabulary, compared to 3.17 of Llama2’s [149] 32k vocabulary. Tao et al. [147] derived scaling laws that show that even this number might be below optimal.

Of course, in large language models with 10s to 100s of billions of parameters, even such a huge vocabulary does not make up a sizable portion of the total parameter budget. But for applications on regular desktops, or even laptops and mobile devices, small models are needed. For example, the smallest model in the Gemma2 [132] series has just 2 billion parameters. With a vocabulary size of 256k and an embedding dimension of 2304, the token embeddings take up 590M, almost 30% of the total model.

Because of this, techniques from XMC have recently found their way into LLM training, such as Liger [60] implementing a skip-loss classification layer that never materializes the logits, similar to the same strategy in Renee [66]. As such, even though next-word prediction might no longer be considered a core XMC problem, it shares enough of the characteristics to enable transfer of useful strategies.

1.3 Current Challenges

Despite many advancements in recent years, extreme multilabel classification remains a challenging problem, both from a statistical and from a computational perspective. This section introduces the particular challenges tackled within this thesis.

Performance measures for long tails The extreme imbalance in data distribution means that estimating the performance of an extreme classifier in a meaningful way is in itself a difficult task. Many common performance measures, such as precision-at- k , are insensitive to the performance of the classifier on tail labels (c.f. Table 4.1). The currently-established way of reporting a tail-label sensitive performance metric is via *propensity-scored* metrics [64]. However, these metrics are derived as unbiased estimates of standard metrics under a certain model of missing labels — only because that model assumes that tail labels go missing with higher probability do these metrics happen to indicate tail-label performance, conflating the two independent issues. Even worse, these metrics contain two *dataset-dependent* hyperparameters, estimated only on two datasets with the recommendation to “[...] are set to their values averaged over Wikipedia and Amazon.” [64], making comparisons across datasets difficult.

This leads to the first question this thesis tries to solve:

Research Question 1a (Long-tail performance metrics). *Can we find (dataset-independent) performance measures that are sensitive to a classifier’s performance on tail labels; in particular, a classifier that never predicts any tail labels should not be able to get a high score on such metrics.*

The natural follow-up, once a performance measure of interest is identified, is to ask what we need to do to perform well on that measure:

Research Question 1b (Optimal predictions). *Can we design algorithms that are consistent for these new performance measures, that is, algorithms that lead to optimal decisions in the limit of infinite training data.*

Due to the enormous label space size in XMC, an optimal algorithm will not be useful in practice if its costs increase rapidly with the number of labels. A refinement of the question above is thus:

Research Question 1c (Efficient approximately-optimal predictions). *Can we design consistent algorithms whose time complexity is no more than linear in the number of labels? If not, are there approximations that can reduce the cost towards linear in the label space?*

Missing Labels As already mentioned above, propensity-scored losses were originally derived as unbiased estimates for existing losses under a specific model of missing labels [64]. However, the analysis is limited to a restricted class of loss functions, those that decompose into a sum of contributions over all *positive* labels. While this covers many losses, in particular the popular precision-at- k , other important losses, notably the binary cross entropy and recall-at- k , do not fall within this framework.

Research Question 2a (Unbiased estimates for general multilabel loss functions). *Can we design unbiased estimates for a larger class of loss*

functions than those covered by Jain et al. [64]? Of particular interest are losses that arise from popular multilabel reductions [110], as these are typically used during model training.

Again, for practical usefulness, unbiasedness is only one of the desirable criteria [24]. Additionally, we need the ability to evaluate these losses quickly, and they should lead to well-defined optimization problems. In particular, unbiased losses are known to lead to increased variance [37], which might make them unsuitable for training.

Research Question 2b (Efficiency of unbiased estimates). *For which losses can unbiased estimates be calculated in linear time in the number of labels? Which losses can be used for stable model training without requiring a much higher number of training samples than their non-unbiased counterparts?*

Finally, in cases where the unbiased versions do not lead to reliable training, one could drop the exact unbiasedness constraint, and instead aim for a more relaxed condition.

Research Question 2c (Surrogates for unbiased estimates). *Can we find surrogate losses for the unbiased versions, that are more amenable to gradient-descent based optimization in the data-scarce XMC regime.*

Efficient Implementations In addition to making sure the newly proposed extensions above remain compatible with the extreme setup, we also want to improve the underlying models themselves.

Linear models still serve as a strong baseline for XMC tasks, and have the convenient property that they lead to embarrassingly parallel training tasks. In theory, that could mean that increasing the number of cores assigned to a training run would linearly decrease the training duration. However, this is not the case on modern hardware. In particular, directly running the parallel `Dismec` [4] training on a 128-core CPU is slower if all 128 cores are used, than if the same training is run using only 32 cores. This discrepancy arises because even though the computations are completely independent, the memory interface is shared between threads, creating a bottleneck.

Research Question 3 (Efficient linear models). *Can we, using knowledge of modern hardware and algorithmic improvements, speed up the training of sparse linear extreme classifiers?*

The independence of individual classifiers in the linear setting means that one can train the classifiers sequentially, or in a distributed manner, and sparsify their weights directly after training. As such, one can avoid storing the dense representation of the full classifier matrix in system memory, and is only required to keep the small subset of classifiers that are currently being trained. Switching from linear to deeper models, unfortunately, breaks the independence, as hidden features are learned for all labels jointly.

The basic strategy of first training a dense classifier and sparsifying the final weights remains valid, but in the deep learning setup, the full memory cost for the entire dense weight matrix of the classifier has to be paid during training. We could try to leverage sparsity already during the training phase. This is a highly challenging prospect, as sparse matrix multiplications are typically much less efficient on GPUs than dense matrix multiplications, leading to much sparsity research being conducted in what is termed “fake sparsity” [91], that is, the sparse matrix is stored in a dense format together with a binary mask. Thus, in order to gain any memory savings, we need to turn to truly sparse training, which requires special attention to be paid to the actual implementation of matrix multiplication, and the storage format of sparse weights. This leads us to:

Research Question 4 (Dynamic Sparse Training). *Can dynamic sparse training (DST) techniques scale to problems at extreme classification level? In particular, can we match both the predictive performance and computational cost of standard dense training, while reducing the memory consumption to that of sparse training?*

1.4 Contributions

The publications collected in this thesis contribute in various ways towards answering the questions posed above. In particular, Publication I addresses RQ 2a for several popular loss functions used in XMC, combining them with techniques designed to adapt to highly-imbalanced data (RQ 1a), and ensuring convergence by proposing convex surrogates (RQ 2c). This is extended in Publication II, which provides unbiased estimates for any loss function derived using commonly employed multilabel reductions [110] (cf. section 2.1); in particular, it covers loss functions that do not decompose over the label space. In the general case, the computational cost is growing exponentially in the number of *positive* labels, not potential labels (RQ 2b), but high variance limits the practical applicability of these losses in the data-scarce XMC regime.

A second limitation is that the unbiased estimates assume knowledge of a model of *propensities* [64], describing the rate at which labels go missing. These rates have been estimates (with some caveats) in Jain et al. [64] for two datasets, and since been applied to other datasets without careful consideration. Publication III presents a critical look at the role of propensity models, including the current practice of using them for both compensating missing labels, but also interpreting propensity-scored metrics as tail-label specific measures of a model’s performance.

Publications VII and VIII then turn to the question of more principled versions of long-tail metrics (RQ 1a). In particular, macro-averaged per-

formance metrics are proposed (cf. section 4.2), which calculate a value for each label individually, and then form the final metric by taking the arithmetic mean over all labels, thus giving each label equal contribution. When switching from loss functions that are defined on the level of a single instance to more general formulations, however, two distinct frameworks arise for formally defining optimal predictions (RQ 1b): the *population-utility* (PU) and *expected test utility* (ETU) frameworks, detailed in section 4.1. As such, Publication VII investigates the ETU setting, developing a *block coordinate ascent* [58, 150] based algorithm to approximate the combinatorial optimization problem inherent in the ETU setting (RQ 1c). Publication VIII proposes to use the Frank-Wolfe [44] algorithm for optimal predictions in the PU setting. Finally, a follow-up that is not part of this thesis, extends these results to cover the online-learning setup [84].

Publications IV, V, and VI tackle the efficiency aspect of XMC algorithms. Publication IV adapts `Dismec` [4] to run efficiently on modern 128-core AMD GPUs, and proposes a new way to initialize the weight vectors based on simple dataset statistics in a way that decreases the overall running time drastically (RQ 3). Publication V serves as a proof-of-concept that dynamic sparse training can be used to learn classifier weights in the extreme setting (RQ 4). In that work, a simplified setting of fixed input features was assumed, i.e., only a linear classifier is trained. This limitation is lifted in Publication VI, which demonstrates that end-to-end training of an extreme classification system using dynamic sparsity is possible, but requires a few additional tricks. In particular, training with a sparse classification layer seems to yield less effective gradients to train the text embedding model, so an auxiliary loss that bypasses the sparse layer becomes necessary.

1.5 Structure of the Thesis

In chapter 2, background relevant to the thesis as a whole is introduced: a more formal specification of the XMC problem and associated loss functions (section 2.1), and an overview over different existing XMC methods (section 2.2). Background that is specific to a single chapter is introduced in a dedicated section of that chapter.

Next, chapter 3 describes the main results of the thesis concerning missing labels, that is, the derivation of unbiased estimators for generic XMC loss functions from Publication I and II, as well as a critical discussion of the use of such estimates in the current XMC literature presented originally in Publication III.

The main point of critique is the dual use of unbiased estimates for both addressing missing labels and long tails. Consequently, chapter 4 discusses the results of Publication VII and VIII towards using and optimizing macro-averaged metrics for the purpose of judging long-tail predictive

performance.

The computational contributions are detailed in chapter 5, and describe improvements made to Dismec [4] from Publication IV, as well as methods to use dynamic sparsity for XMC training proposed in Publication V and VI.

2. Preliminaries

2.1 Task Losses and Multilabel Reductions

Before describing concrete XMC models, we first need to introduce the specific tasks we want to solve, by defining *task losses*¹ that need to be optimized. As these are generally non-differentiable, they cannot be handled with standard gradient-descent methods, and instead, differentiable convex *surrogate* losses need to be defined, usually by means of a multilabel reduction.

Setup and Notation Let $\mathbf{x} \in \mathcal{X}$ denote a data point (e.g., the content of a Wikipedia article), then the task of an XMC model ϕ is to map that point to a set $\hat{\mathcal{P}} \subset [m]$,² $\phi: \mathbf{x} \mapsto \hat{\mathcal{P}}$. For notational convenience, we usually represent label sets through their indicator vector $\hat{\mathbf{y}} \in \{0, 1\}^m$ such that $\hat{y}_i = 1 \Leftrightarrow i \in \hat{\mathcal{P}}$. We assume that instances and their corresponding labels are distributed according to some joint distribution $(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}$, where we use capital letters to indicate that \mathbf{X} and \mathbf{Y} are random variables.

Bayes Classifier and Consistent Algorithms What makes a good classifier? This is task-dependent, and is encoded in a classifier's *risk* $R_\ell[\phi, \mathbb{P}]$. In this chapter, we will consider only simple risks that can be written as an expectation of a loss function ℓ over individual instances sampled i.i.d. from the data distribution \mathbb{P} :

$$R_\ell[\phi, \mathbb{P}] = \mathbb{E}_{\mathbf{X}, \mathbf{Y} \sim \mathbb{P}} [\ell(\phi(\mathbf{X}), \mathbf{Y})]. \quad (2.1)$$

A good classifier among a family of candidates \mathcal{H} is characterized by having low risk; in particular, the optimal — or *Bayes* — classifier ϕ^* for a given

¹In slight abuse of terminology, we use the term loss for any quantity that we want to optimize for, whether it is by minimization (loss) or by maximization (utility).

²We denote with $[m] := \{1, 2, \dots, m\}$ the set of integer up to m , and with $\mathbb{1}[\cdot]$ the indicator function.

data distribution and loss function is the one that minimizes the risk³

$$\phi^* := \operatorname{argmin}_{\phi \in \mathcal{H}} R_\ell[\phi, \mathbb{P}]. \quad (2.2)$$

In the case of decomposable risk as described above, the Bayes-classifier can be computed point-wise, and is given by

$$\phi^*(\mathbf{x}) = \mathbf{y}^* = \operatorname{argmin}_{\hat{\mathbf{y}}} \mathbb{E}[\ell(\hat{\mathbf{y}}, \mathbf{Y}) | \mathbf{X} = \mathbf{x}]. \quad (2.3)$$

In chapter 4, we will consider a setup in which this no longer holds.

Of course, the actual data distribution \mathbb{P} is not available to us. Instead, we can only calculate an *empirical* version of the risk over a finite dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$ through

$$\hat{R}_\ell[\phi, \mathcal{D}] := n^{-1} \sum_{i=1}^n \ell(\phi(\mathbf{x}_i), \mathbf{y}_i), \quad (2.4)$$

and try to choose a classifier that minimizes (a regularized version of) the empirical risk.

Task losses for XMC This definition of risk still leaves open the choice of loss function. One option is to check whether the two vectors are identical

$$\ell_{\text{SS01}}(\hat{\mathbf{y}}, \mathbf{y}) := \mathbb{1}[\mathbf{y} \neq \hat{\mathbf{y}}] = 1 - \prod_{i=1}^m \mathbb{1}[\mathbf{y}_i = \hat{\mathbf{y}}_i]. \quad (2.5)$$

This *subset zero-one loss* is not suitable for XMC tasks, however, as the likelihood of *all* entries being correct in the label vector becomes very small as the vector size increases to hundreds of thousands of entries.

On the other extreme of potential loss functions is the *Hamming loss*, defined as

$$\ell_{\text{Ham}}(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{m} \sum_{i=1}^m \mathbb{1}[\hat{\mathbf{y}}_i \neq \mathbf{y}_i], \quad (2.6)$$

the fraction of labels classified incorrectly. This loss is also unsuitable for XMC, because of the extreme class imbalance. Even a trivial classifier that predicts $\hat{\mathbf{y}} \equiv \mathbf{0}$ everywhere would get almost perfect Hamming-loss.

Instead, XMC losses are typically derived by exploiting that usually, XMC methods produce a *ranking* over possible labels by assigning a real-valued score to each label, and then predict the highest-ranked labels. Selecting a fixed number k of labels to predict, i.e. restricting $\|\hat{\mathbf{y}}\| = k$, one can define *precision-at-k* and *recall-at-k* through

$$\text{P@k}(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{k} \sum_{i=1}^m \mathbf{y} \cdot \hat{\mathbf{y}}, \quad \text{R@k}(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{\|\mathbf{y}^*\|_1} \sum_{i=1}^m \mathbf{y} \cdot \hat{\mathbf{y}}. \quad (2.7)$$

³We tacitly assume that such a classifier exists; in full generality, one can only define the Bayes risk as the infimum of the risk over all classifiers, and consider sequences of classifiers whose risk converges towards the Bayes risk.

Note that, contrary to traditional precision and recall metrics, which admit trivial classifiers that predict nothing or everything as optimal, respectively, the “at- k ” constraint does not allow such trivial solutions. To make this more explicit, we can instead formulate these losses to operate on the scores directly, e.g.,

$$\text{P@k}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{k} \sum_{j=1}^k y_{\text{arg}_j(\hat{\mathbf{y}})}, \quad (2.8)$$

where $\text{arg}_j(\hat{\mathbf{y}})$ denotes the index of the j^{th} largest entry in $\hat{\mathbf{y}}$.

With a ranking of labels available, we might want to put more emphasis on the highly ranked labels. This is achieved with *discounted cumulative gain* (DCG) [68]

$$\text{DCG@k}(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{k} \sum_{j=1}^k \frac{y_{\text{arg}_j(\hat{\mathbf{y}})}}{\log(j+1)}, \quad (2.9)$$

or its normalized variant

$$\text{nDCG@k}(\hat{\mathbf{y}}, \mathbf{y}) := \frac{\text{DCG@k}(\mathbf{y}^*, \hat{\mathbf{y}})}{\sum_{j=1}^{\min(k, \|\mathbf{y}^*\|_1)} \log(j+1)}. \quad (2.10)$$

Additionally, given the long-tailed label distribution in XMC, an upweighting of tail labels using propensity scores $\mathbf{p} \in (0, 1]^m$ might be desired, leading to *propensity-scored* precision

$$\text{PSP@k}(\hat{\mathbf{y}}, \mathbf{y}) := \frac{1}{k} \sum_{j=1}^k \frac{y_{\text{arg}_j(\hat{\mathbf{y}})}}{p_j}, \quad (2.11)$$

with analog definitions for propensity-scored recall and DCG. An important caveat is that, while publications usually provide the formula as in (2.11), the actual numbers reported are in fact normalized⁴ to ensure that they lie between zero and one, through

$$\text{nPSP@k}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\text{PSP@k}(\hat{\mathbf{y}}, \mathbf{y})}{\text{PSP@k}(\mathbf{y}, \mathbf{y})}. \quad (2.12)$$

All losses above have in common that they require the transformation of continuous scores into discrete predictions, making them not directly learnable via gradient descent. This problem is not specific to XMC; binary task losses such as precision, accuracy, recall, or F1 measure are also non-differentiable. This is typically addressed by means of *surrogate losses*, convex and differentiable functions which when optimized lead to a minimum that also optimizes the corresponding task loss [6]. Consider a sequence of classifiers ϕ_n that minimizes the empirical surrogate risk w.r.t. $\tilde{\ell}$ as the dataset size goes towards infinity,

$$\phi_n \in \underset{\phi \in \mathcal{H}}{\text{argmin}} \hat{R}_{\tilde{\ell}}[\phi, \mathcal{D}_n] \quad \text{for } \mathcal{D}_i = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\} \sim \mathbb{P}^n, \quad (2.13)$$

⁴A more elaborate discussion of this problem, and the dual role of PSP@k as a tail-label metric as well as an unbiased estimate under missing labels, see section 3.4.

then we want

$$\lim_{n \rightarrow \infty} R_\ell[\phi_n, \mathbb{P}] \rightarrow \inf_{\phi \in \mathcal{H}} R_\ell[\phi, \mathbb{P}]. \quad (2.14)$$

In that case, optimizing the surrogate $\tilde{\ell}$ is *consistent* for optimizing the task loss ℓ .

While it is possible to derive surrogates for XMC losses directly, the more common approach is to map the multilabel problem into multiclass or binary problems, for which a rich literature on surrogate losses already exists.

Multilabel reductions For example, one could map the 2^m different binary vectors of possible combinations of labels to one class each, $\mathbf{y} \mapsto y_1 + 2y_2 + \dots + 2^{m-1}y_m$, and then solve a 2^m -class multiclass problem. Of course, this transformation is completely intractable at the extreme label scale.

Instead, the most commonly used *multilabel reduction* in XMC is *one-vs-all* (OVA),⁵ in which the m -label multilabel problem is decomposed into m separate binary problems, so that surrogates such as the (squared) hinge loss and binary cross-entropy can be applied.

Similarly, one can transform the multilabel problem into a multiclass problem as follows: From any given data point $(\mathbf{x}, \mathcal{P})$ with label set $\mathcal{P} = \{p_1, \dots, p_t\}$, we create t new data points $\{(\mathbf{x}, p_1), \dots, (\mathbf{x}, p_t)\}$ that have exactly one label each. This allows model training to employ categorical cross-entropy as a surrogate loss. This is called the *pick-all-labels* (PAL) reduction. Instead of picking all labels, one could also uniformly select one (*pick-one-label*, POL) as the multiclass target.

It turns out that these two variations lead to models that converge to different optima: Whereas PAL and OVA are consistent for precision-at- k and Hamming task-loss, POL is not [162].⁶ On the other hand, POL, and normalized versions of OVA and PAL, do lead to consistent classifiers with regard to the recall-at- k metric [110].

Two-stage prediction Another perspective on optimizing task losses can be gained by splitting the inference procedure into two parts. If we can show that for determining the optimal prediction, the vector of label probabilities $\eta_j(\mathbf{x}) = \mathbb{P}[Y_j = 1 \mid \mathbf{X} = \mathbf{x}]$ is a sufficient statistics, i.e., we can write $\mathbf{y}^*(\mathbf{x}) = \mathcal{I}(\boldsymbol{\eta}(\mathbf{x}))$ for some inference algorithm \mathcal{I} , we can reduce the problem of optimal decisions under the task loss to the problem of estimating $\boldsymbol{\eta}$. This strategy is used in Publications VII and VIII to derive consistent classifiers that optimize *macro-averaged* losses, cf. chapter 4.

⁵Also called one-vs-rest

⁶Except in the special case when labels are independent.

2.2 XMC methods

After the task is defined, a wide variety of models can be considered. These include shallow models like linear classifiers, deep models based on transformer networks, and models designed to exploit *label features*.

Linear Models Potentially the simplest method for extreme classification is to use a large linear classifier to produce prediction scores. This requires to embed the input text documents into a vector space. These can be bag-of-word representations like *tf-idf* [57]. Even though these purely statistical features ignore any information about word ordering, they have long performed on par with or better than deep-learning based approaches. This has been attributed to the extreme data-scarcity affecting XMC methods [130], and thus only with the arrival of transformer models pre-trained on huge amounts of non-XMC data (to be discussed below) have linear methods lost the prime position among XMC classifiers.

Naïvely, it might seem that this approach would be completely intractable. For a vocabulary size of 100000 terms in the documents, and one million labels, the resulting classifier matrix would have a total of 100 *billion* parameters, corresponding to 400 gigabytes of memory in single precision. There are two important insights that keep this method possible: First, when using the one-vs-all reduction, the problem is turned from a single one-million-parameter task into one million *independent* binary classification problems. Second, only a small subset of document terms will be relevant to any given label, so that the classifier weights can be highly sparse. This is the key to methods like *Dismec* [4] and *ProXML* [5], which train L_2 and L_1 regularized linear classifiers, respectively, in a massively parallel fashion.

Still, training such large linear classifiers remains computationally costly. As we show in Publication IV, a careful implementation adapted to the underlying hardware, along with a clever initialization scheme, can still provide significant speed-ups to the methods while yielding identical results (cf. section 5.1).

Because the average number of labels relevant to a document, and the average number of documents relevant to a label, are both small compared to the problem size in XMC (cf. Table 1.1), it can be shown that a primal-dual formulation of optimizing the max-margin loss leads to highly sparse solutions [171, 170].

Combined with hierarchical label clusters (described below), linear models can even be scaled to millions of labels [174].

Shallow Embeddings In (shallow) embedding models, the input features are projected to a lower-dimensional vector space, before applying linear classification in the extremely large label space. Embedding approaches include *S1eec* [12], *Annexml* [146], and *Lem1* [173]. Initially, it was believed that *linear* embedding methods cannot be successful, because they implicitly

assume a low-rank structure in the label matrix that is violated in real-world data [12]; however, Guo et al. [51] showed that, in fact, linear embedding methods are sufficiently expressive as long as the number of relevant labels for each sample can be bounded by a small number.

Shortlisting and Label Trees In order to avoid the linear growth of compute cost with the label space size, several tree-based and shortlisting methods have been devised. For example, *probabilistic label trees* (PLT) [69, 70] build a tree data structure in which the leaf nodes are the labels, and decompose the computation of label probability according to the chain rule [10]. In this way, PLT can be seen as a generalization of the *hierarchical softmax* [117] to the multilabel case [162]. During training, only the nodes along the paths leading to ground-truth labels need to be visited, and during inference, efficient search algorithms like beam search or A* can be used to achieve sublinear compute cost [87, 163].

That still leaves the question of constructing the label tree. In Parabel [127] and many follow-up works, first each label is associated with a feature vector by taking the average of features of all positive instances for this label. Based on this representation, a hierarchical k-means clustering procedure then produces the label tree. Bonsai [78] improve upon Parabe by using shallow trees with a large branching factor, which results in more accurate classification.

Instead of exactly modeling the factorized probability, one can also take the view that, in the presence of a large amount of easy negatives, many loss functions can be approximated by a small subset of contributions, positives and hard negatives:

$$\ell(\hat{\mathbf{s}}, \mathbf{y}) = m^{-1} \sum_{i \in \mathcal{D}} (\ell(\hat{s}_i, 1)) + m^{-1} \sum_{i \in [m] \setminus \mathcal{D}} (\ell(\hat{s}_i, 0)) \quad (2.15)$$

$$\approx m^{-1} \sum_{i \in \mathcal{D}} (\ell(\hat{s}_i, 1)) + m^{-1} \sum_{i \in \mathcal{N}_+} (\ell(\hat{s}_i, 0)), \quad (2.16)$$

where the set \mathcal{N}_+ of hard negatives is chosen in a way that tries to minimize the difference between (2.15) and (2.16). This choice might be guided by label-trees, too. At each level of the tree, the classifier solves a coarse-grained version of the XMC problem, where a node is labeled as a positive if *any* of its descendants lead to a positive leaf node. The short-list can then be formed by selecting only the highest-scoring of these intermediate nodes.

Note that, in contrast to the PLT technique above, the predicted scores for each shortlisted label are determined *purely* by the classification at the last layer, and any label not in the shortlist has a score of $-\infty$.

Alternatively, the shortlist can be constructed using some form of approximate maximum inner product search (MIPS), e.g., through *locality-sensitive hashing* (LSH) [49, 155, 21]. This is straightforward for inference, when the corresponding data-structures can be pre-computed offline, but during

training the weight vectors change. Therefore, one needs to find a balance computational cost of frequent hash updates against the quality cost associated with stale shortlists [107, 129]. Another challenge is that the gap between logits of positive and the highest-ranked negative labels might be small, making it difficult for LSH to ensure good recall of positives [98].

In addition to elaborate shortlisting schemes, simple heuristics can be used, too, such as sampling negatives uniformly or according to some fixed distribution based on label frequencies. From the computational perspective, the most attractive scheme is *in-batch negative mining*, where the positive labels for other instances in a batch are used as negatives during that batch. By careful construction of training batches, as opposed to i.i.d. sampling, one can choose batches so that in-batch negative mining serves as hard-negative mining [28].

Most modern methods for XMC combine some form of shortlisting with multiple layers of learned features, as described below.

Deep-Learning Methods Early attempts at using deep learning methods for XMC include the convolution-based XML-CNN [95]⁷ and the LSTM-based AttentionXML [172]. However, it is only with the advent of large pre-trained Transformer models [154] like Bert [32] or XLNet [166] that deep methods have started to supplant simpler models.

The first methods to fine-tune such large pretrained Transformer models were X-Transformer [19] and APLC-XLNet [168]. Both methods employ clustering of labels to reduce the computational cost. X-Transformer uses the clusters also to achieve a more balanced training objective for its separately-trained submodules, whereas APLC-XLNet performs end-to-end training with a PLT formulation of the cross-entropy loss. LightXML [72] learns the shortlisting module jointly with the extreme classifier, and XR-Transformer [177] extends X-Transformer to use a multi-level tree. CascadeXML [79] brings this two improvements together with end-to-end learning of a multi-level classifier that extracts classification features for coarse-grained label clusters from earlier layers of the Bert model. Despite the success of Transformer models, it still proves helpful to combine the Transformer-based representations with the simpler tf-idf representation for best results [19, 177, 79].

A general framework for modular deep extreme classification is introduced in Dahiya et al. [27], which proposes a 4-stage process: First, deep embeddings are trained by means of an auxiliary task. Then, these preliminary embeddings are used to generate an indexing data structure, which in turn can be used to generate a shortlist of hard negative labels for each training instance. Finally, the extreme classifier itself is trained jointly with fine-tuning the embeddings. Care needs to be taken to ensure the fine-tuned embeddings remain close to the original ones, so that the

⁷Note that there is a significant discrepancy between the algorithm as described in the paper, and its reference implementation [22]

precomputed shortlists remain meaningful. If each of these steps can be performed in $\mathcal{O}(\log m)$ time, then the entire method scales to even many millions of labels, and many different instantiations of this blueprint can be implemented [113, 26, 136, 113].

While the increase in available hardware resources, as well as improved implementation of key components such as the loss calculation, have made it possible to train extreme classifiers with deep-learning backends fully end-to-end without any shortlisting [66], this requires long training times, so sublinear approaches are still highly desirable [107].

Label Metadata In order to alleviate the lack of training data, especially for tail labels, recent approaches have started to take into account additional label metadata, instead of treating labels as featureless integers. Examples of such metadata include textual descriptions of labels (*label features*) or the label co-occurrence matrix. Co-occurrence statistics can be used, for example, to constrain the Gram matrix of classifier weights to reflect the relative frequencies of labels co-occurring with each other [51].

Textual label features are of particular relevance in short-text XMC settings; datasets in which the training points themselves are only short phrases of a few words, like the titles of a website, product, or Wikipedia article, or a user’s search query [27]. In that case, the paucity of available training data, also in the input texts, makes *frugal* architectures a competitive alternative to transformers [27, 113, 80]. As typical label descriptors are also short phrases, this often leads to a symmetric learning problem in which label and document features are interchangeable. This enables the use of Siamese [138] architectures such as SiameseXML [26] and NGame [28], as well as data-augmentation strategies [81, 23].

Related Settings While most works dealing with extremely large output spaces, including this thesis, focus on the classification setting, such problems also arise in other domains.

In the bandit setup, the output space is given by the set of possible arms, and instead of a labelled dataset, the machine learning system has to make inferences purely based on the reward given for selected arms. If the arms correspond to, e.g., possible advertisements to present to a user, out of a pool of millions of ads, and the only feedback is whether the user clicked on the presented ad (no information is available for non-presented ads), then one has to do “Learning from Extreme Bandit Feedback” [100]. Other works dealing with extreme action spaces include Sen et al. [139], Saito and Joachims [137] and Sachdeva et al. [134]. Other recent developments

are *zero-shot* extreme classification [52, 165, 178, 175] (generalization to unseen *labels*), multimodel XMC Mittal et al. [114], and regression [128]. Note that many XMC methods implicitly solve the regression problem of estimating the conditional probabilities η of labels given the input text, even if the corresponding task losses do not take these probabilities into account.

3. Missing Labels

This chapter presents the contributions of Publication I, II, and III towards addressing missing labels in extreme multilabel classification. The main results of I and II are unbiased estimators and surrogates for generic one-vs-all losses (I), as well as pick-all-labels and the corresponding normalized reductions (II). Note that the original publications do not use the same notation consistently; thus the equations presented here may not be verbatim copies, but are instead cast in a common notation.

3.1 Background: Noisy- and Missing-Label Learning

As discussed in the introduction, the sheer scale of XMC problems makes it impossible to collect carefully human-annotated datasets. Without any additional assumptions, however, it is impossible to make any progress on that task. To formalize the setting, let \mathbf{Y} denote the true label vector, and $\tilde{\mathbf{Y}}$ the observed (i.e., noisy) labels, defined on a joint probability space and distributed according to $\mathbb{P}_{\mathbf{Y}}$ and $\mathbb{P}_{\tilde{\mathbf{Y}}}$ respectively. Then, in their most general form, noisy labels can be described as a transformation of one distribution into the other, either as $\mathbb{P}_{\tilde{\mathbf{Y}}} = t(\mathbb{P}_{\mathbf{Y}})$ in the instance-independent case, or $\mathbb{P}_{\tilde{\mathbf{Y}}|\mathbf{x}} = t(\mathbb{P}_{\mathbf{Y}}, \mathbf{x})$ if the noise is instance-dependent, where t is a transition function.

General theory for learning with corrupted labels As $\mathcal{Y} \ni \mathbf{Y}$ is a finite set, a distribution $\mathbb{P}_{\mathbf{Y}}$ can be identified by the vector of probabilities \mathbf{p} for each element in that set. If t can be represented as an invertible linear map $\mathbb{T} \in \text{GL}(2^m)$, i.e., $\mathbb{P}_{\tilde{\mathbf{Y}}} = \mathbb{T}\mathbb{P}_{\mathbf{Y}}$, then Van Rooyen and Williamson [152, Theorem 5] show that, for *any* function $f: \mathcal{Y} \rightarrow \mathbb{R}$, there exists an unbiased version \tilde{f} such that

$$\forall \hat{\mathbf{y}}: \mathbb{E}[f(\hat{\mathbf{y}}, \mathbf{Y})] = \mathbb{E}[\tilde{f}(\hat{\mathbf{y}}, \tilde{\mathbf{Y}})]. \quad (3.1)$$

The argument is quite simple, albeit abstract: Gathering all possible values of $f(\hat{\mathbf{y}}, \mathbf{Y})$ in \mathbf{f} , the expectation $\mathbb{E}[f(\mathbf{Y}, \hat{\mathbf{y}})]$ can be written as an inner product $\langle \mathbf{f}, \mathbf{p} \rangle$. Similarly, in the noisy case we have $\langle \tilde{\mathbf{f}}, \mathbb{T}\mathbf{p} \rangle$. To make these equal,

consider

$$\langle \mathbf{f}, \mathbf{p} \rangle = \langle \mathbf{f}, \mathbb{T}^{-1} \mathbb{T} \mathbf{p} \rangle = \langle \mathbb{T}^{-\dagger} \mathbf{f}, \mathbb{T} \mathbf{p} \rangle, \quad (3.2)$$

where $\mathbb{T}^{-\dagger}$ is the adjoint of the inverse of the corruption operator. Thus $\tilde{\mathbf{f}} = \mathbb{T}^{-\dagger} \mathbf{f}$ is the unbiased version of \mathbf{f} , which exists for any invertible corruption \mathbb{T} .

While this is a fully general statement, it does not give much insight into more specific cases of interest, and implementing it as-is would require inverting a potentially very large corruption matrix \mathbb{T} .

Learning from noisy, binary labels For the specific case of noisy labels in binary classification, Natarajan et al. [122, 123] show that with noise rates ρ_{+1} and ρ_{-1} for positive and negative instances, respectively, any function $f: \{0, 1\} \rightarrow \mathbb{R}$ can be made unbiased by

$$\tilde{f}(\hat{y}, y) = \frac{(1 - \rho_{-y})f(\hat{y}, y) - \rho_y f(\hat{y}, -y)}{1 - \rho_{+1} - \rho_{-1}}, \quad (3.3)$$

where $y \in \{+1, -1\}$ denotes a binary label, and $\rho_{\pm 1}$ are the noise rates for positives and negatives, respectively.

Setting $\rho_{-1} = 0$ (i.e., negative labels remain unchanged), this tells us that in the missing-labels setting,

$$\tilde{f}(\hat{y}, 1) = \frac{f(\hat{y}, 1) - \rho_{+1} f(\hat{y}, -1)}{1 - \rho_{+1}}, \quad \tilde{f}(\hat{y}, -1) = \frac{(1 - \rho_{+1})f(\hat{y}, -1)}{1 - \rho_{+1}}. \quad (3.4)$$

Jain et al. [64] define the *propensity*

$$p := \mathbb{E}[\tilde{Y} | Y = 1] = 1 - \rho_{+1} \quad (3.5)$$

as the rate of a label *not* going missing, and additionally assume that loss functions used in XMC fulfill $f(\hat{y}, -1) = 0$, that is, they focus on the positive labels. Summing up individual binary losses in an OVA decomposition yields

$$\tilde{f}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{j=1}^m \frac{f(\hat{y}_j, y_j)}{p_j}, \quad (3.6)$$

which can be used to define unbiased estimators for the precision-at- k metric and is in widespread use under the name of propensity-scored precision-at- k [11]. However, other typical loss functions, like recall-at- k , do not admit an OVA decomposition, and thus are not directly addressable using (3.6).¹

In addition to considering Y and \tilde{Y} as two separate random variables constrained by the probabilistic model of the noise, one can also make their relation algebraic by introducing *mask variables* [148]

$$\tilde{Y} = M \cdot Y. \quad (3.7)$$

where $M \sim \text{Ber}(p)$ are Bernoulli-distributed.

¹Jain et al. [64] provide an unbiased estimator, but that requires a-priori knowledge of the *number* of relevant labels for each instance.

Related Settings Two settings closely related to the missing-labels problem are *Positive-Unlabeled* learning [38, 35, 7] and off-policy estimation, presented below.

Positive-unlabeled learning can be considered in two variations. The *censoring* setting is just another name for missing labels. There is a single dataset that was sampled i.i.d. from some distribution, but some of the positives are not marked as such (i.e., they are censored). The *case-control* setting, in contrast, allows for the positive examples to be generated by a process distinct from the unlabeled data. In particular, this means that the ratio of positives to negatives in the training data differs from that in the test data, yielding another parameter that needs to be estimated [38].

In off-policy estimation, one wants to calculate some statistics under a certain data distribution, having available only data gathered with a different distribution. For example, in Joachims et al. [73], implicit user feedback is collected based on click data, in a batch-learning from logged bandit feedback setup [144, 142]. As in each interaction, only a small subset of candidates will be available to the user to click on, leading to similar missing labels problems as in the XMC setup. However, a crucial difference is that the *logged actions* (i.e., whether a candidate was presented) are known. Thus, in off-policy learning, one has available the values of the mask variables \mathbf{M} defined above, whereas these are unknown in a pure missing labels setup. Thus, the equivalent to (3.6) looks like

$$\tilde{f}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{j=1}^m \frac{M_j}{p_j} f(\hat{y}_j, y_j) \quad (3.8)$$

Problems with unbiased estimators While unbiasedness is a desirable property for an estimator, it is not the only criterion that determines whether the estimator is useful. If we plan on using it as an optimization objective, it should be lower-bounded and ideally also convex.

Unfortunately, even if the original quantity fulfills all these properties, its unbiased estimator often does not [82, 34, 24]. Natarajan et al. [122] show that a sufficient condition for a convex unbiased version of a binary loss function is $\partial_{\hat{\mathbf{y}}}^2 f(\hat{\mathbf{y}}, \mathbf{y}) = \partial_{\hat{\mathbf{y}}}^2 f(\hat{\mathbf{y}}, -\mathbf{y})$. However, this condition does not guarantee lower-boundedness.

An additional problem of unbiased estimators, and (inverse) propensity-scored estimators in particular, is that they can suffer from high variance [37]. This is particularly problematic when using these estimates as optimization objectives, where it can lead to *propensity overfitting* [145].

Alternatives to unbiased estimators While this chapter focuses on the method of unbiased estimators to deal with the missing-labels problem, there are other approaches worth mentioning:

If one chooses the right loss function, and the label noise is sufficiently benign, it might be possible that doing *nothing* is actually sufficient; in this

case, the loss function is *robust* to noise. This happens, for example, in binary classification with noise rates of less than 50% if the loss function fulfills a symmetry condition $\ell(\cdot, 0) + \ell(\cdot, 1) = c$ for some constant c [48, 153]. Other examples include the balanced error or the AUC [109], but typical convex losses are not robust [99].

Instead of trying to adapt the learning process to the presence of noise, another approach is to try and filter the training data in an attempt to remove mislabeled instances [54, 71, 164]. In the XMC setting, the soft labels used in Lever [15] and Gandalf [81] could be seen in this light, though they are not motivated purely from a missing-labels perspective.

Finally, if the model is designed to output a probability distribution over labels, and the noise rates are known as in the unbiased setting, then one can also just train the model to produce the corrupted probabilities as outputs, and use the inverse of the noise model to convert to ground-truth probabilities only during the inference procedure [163, 126].

3.2 Unbiased estimates and their surrogates for One-vs-All loss functions

The results derived in Jain et al. [64], equation (3.6), allow for calculating unbiased estimates for common task losses such as precision-at- k , but due to the requirement that $\ell(\hat{y}, 0) = 0$, loss functions used during training, such as binary cross-entropy or hinge loss, cannot be treated with this equation.

Thus, Publication I presents a corollary of Natarajan et al. [122, Lemma 1], giving the form of the unbiased version of a generic binary loss function under missing labels as

$$\tilde{\ell}(\hat{y}, y) = \begin{cases} p^{-1}(\ell(\hat{y}, 1) + (p-1)\ell(\hat{y}, 0)) & y = 1 \\ \ell(\hat{y}, y) & y = 0 \end{cases}. \quad (3.9)$$

However, functions thus defined turn out to be ill-suited for model training, as they can be not lower-bounded due to the $(p-1)\ell(\hat{y}, 0)$ term if $\ell(\hat{y}, 0)$ is not upper-bounded. To address this problem, one approach is to remember that these losses can be interpreted as surrogates to the non-differentiable 0-1 loss. As demonstrated in Figure 3.1, the order of operations, that is, of forming a convex upper-bound and of forming the unbiased version, matters. In particular, forming an unbiased estimate of the 0-1 loss results in a bounded function, and taking a convex upper-bound surrogate of that guarantees that the constructed function is both lower-bounded and convex, and thus a good candidate for an optimization objective.

To follow common convention, one can shift the resulting function so that it results in zero loss for classifications that are correct with sufficient margin. In fact, the method presented above is equivalent to the method of label-dependent costs from Natarajan et al. [122, section 4].

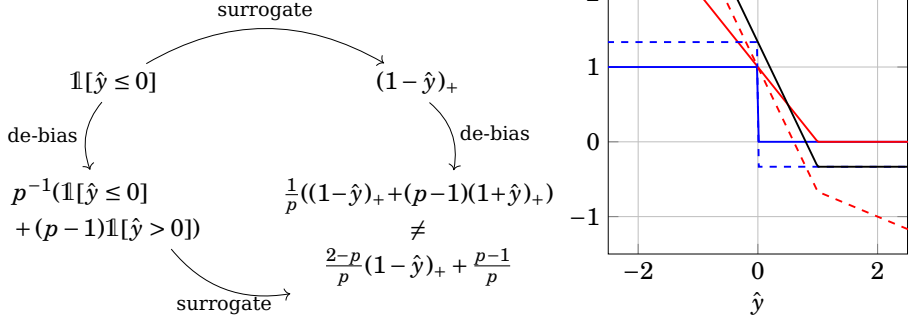


Figure 3.1. Non-commutativity of taking the unbiased estimate and forming a surrogate by a piecewise-linear function (hinge-loss). The transformations for the positive part of the loss, $\ell(1, \cdot)$, are illustrated on the left side, the graph on the right visualizes these formulas for $p = 0.75$. The 0-1 loss and corresponding unbiased version are shown in solid and dashed blue, respectively; The hinge loss and its unbiased version are solid and dashed red. The black line indicates first taking the unbiased version of the 0-1 loss, and then matching a piecewise-linear function with a margin of 1 as an upper bound.

In addition to compensating missing labels, XMC methods might also want to explicitly address the class imbalance, e.g., through dataset re-sampling [40] or augmentation [20], as well as through adaptations of the loss function [25, 16, 111, 133]. In Publication I, we focus on the latter, assuming to have a generic weighting factor $c^+(n_j, n)$ that is a function of the number of positive samples n_j for a given label, and the total number of samples n .

For a given binary loss function ℓ , the corresponding reweighted one-vs-all version is thus given by

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{j=1}^m (\mathbb{1}[y_j = 1] c^+(n_j, n) + \mathbb{1}[y_j = 0]) \cdot \ell(\hat{y}_j, y_j). \quad (3.10)$$

As above, there are again two possibilities of applying the transformation that given different results: Either apply re-balancing of classes first, and then adapt to missing labels, or vice versa. For the 0-1 loss, these two orders of operations yield

$$\tilde{\ell}(\hat{y}, 1) = c^+(n_j, n)(2p - 1)\mathbb{1}[\hat{y} < 0] \quad (3.11)$$

$$\tilde{\ell}(\hat{y}, 1) = (c^+(n_j/p, n)/p + 1/p - 1)\mathbb{1}[\hat{y} < 0], \quad (3.12)$$

where we used the shifted version of the unbiased 0-1 loss. The first version allows the loss function to be written down in a particularly simple form,

$$\tilde{\ell}_j(\hat{y}_j, y_j) = \begin{cases} w_j^+ c_j^+ \ell(\hat{y}_j, 1) & y_j = 1 \\ \ell(\hat{y}_j, 0) & y_j = 0 \end{cases}. \quad (3.13)$$

When choosing Cui et al. [25] as the rebalancing method, this results in

$$w_j^+ = \frac{2}{p_j} - 1, \quad c_j^+ = \frac{1 - \beta}{1 - \beta^{n_j}}. \quad (3.14)$$

Empirically, we find that when training deep networks, directly applying these scaling factors can result in unstable model training. Therefore, an additional normalization step is applied.

Tables 3, 4, and 5 in Publication I show the results of applying these loss functions for Dismec [4], AttentionXML [172], and APLX-XLNet [168].

3.3 Unbiased estimates and their surrogates for general multilabel reductions

While Publication I gives a comprehensive treatment to loss functions derived using the one-vs-all reduction, the other reductions presented in Menon et al. [110] (cf. section 2.1) are still left open.

In order to make progress here, Publication II reformulates the problem in two ways: First, observed labels are described as the product of a mask variable and the unobserved ground truth labels²

$$\tilde{\mathbf{Y}} = \mathbf{M} \odot \mathbf{Y}. \quad (3.15)$$

This is convenient, because under the standard assumption that the mask \mathbf{M} is independent of labels \mathbf{Y} and instances \mathbf{X} ,³ it can be pulled out of expectations and be replaced by $\mathbb{E}[\mathbf{M}] = \mathbf{p}$.

The second insight is that *any* arbitrary multilabel loss function $\ell: \mathbb{R}^m \times \{0,1\}^m \rightarrow \mathbb{R}$ can be cast as a linear function in its second argument, due to the discrete nature of the observed labels. The rewriting is done in two steps. First, note that due to the finite set of possible observed labels, we can always write

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{\mathbf{y}' \in \{0,1\}^m} \mathbb{1}[\mathbf{y}' = \mathbf{y}] \ell(\hat{\mathbf{y}}, \mathbf{y}'). \quad (3.16)$$

Next, note that an indicator function of binary variables can be written as a linear function

$$\mathbb{1}[y = y'] = y' \cdot y + (1 - y')(1 - y), \quad (3.17)$$

and the vector-valued indicator can be written as a product of element-wise indicators.

For the binary case, the linearization looks as follows:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = y(\ell(\hat{\mathbf{y}}, 1) - \ell(\hat{\mathbf{y}}, 0)) + \ell(\hat{\mathbf{y}}, 0). \quad (3.18)$$

In this form, reading off the unbiased loss function becomes straightforward:

²For the binary case, this has been considered also in Teisseyre et al. [148].

³Note that this is a non-trivial statement; Noise of the missing-label variety is clearly *label-dependent*, but because we multiply with the ground-truth labels, the value of M_j does not matter whenever $Y_j = 0$, so we can *choose* a random-variable \mathbf{M} that fulfills the desired independence.

$$\begin{aligned} \mathbb{E}[\tilde{\ell}(\hat{\mathbf{Y}}, \tilde{\mathbf{Y}})] &= \mathbb{E}[MY(\ell(\hat{\mathbf{Y}}, 1) - \ell(\hat{\mathbf{Y}}, 0)) + \ell(\hat{\mathbf{Y}}, 0)] = \\ &= \mathbb{E}[M] \mathbb{E}[Y(\ell(\hat{\mathbf{Y}}, 1) - \ell(\hat{\mathbf{Y}}, 0)) + \ell(\hat{\mathbf{Y}}, 0)] = \\ &= p \mathbb{E}[Y(\ell(\hat{\mathbf{Y}}, 1) - \ell(\hat{\mathbf{Y}}, 0)) + \ell(\hat{\mathbf{Y}}, 0)]. \end{aligned} \quad (3.19)$$

Thus, the unbiased version can be formed by replacing any occurrence of Y with Y/p . In the multilabel case, the linearized expression can contain products of labels, and correspondingly, products of masks. In that case, it is further required that the individual components of the mask are independent, that is, whether label j goes missing has no influence on whether label j' goes missing.

With these tools, Publication II derives additional results in the binary and decomposable multilabel case:

Generalization Theorem 9 presents Rademacher-complexity based generalization bounds. The statement is a corrected version of Natarajan et al. [123, Lemma 8], which takes into account that the unbiased estimate can have a different range than the original function, see Publication II, p.42.

Uniqueness It can be shown (Theorem 10) that the derived unbiased estimator is essentially unique, in the sense that *any* unbiased estimator is of the form $\tilde{\ell}'(\hat{\mathbf{y}}, \mathbf{y}) = \tilde{\ell}(\hat{\mathbf{y}}, \mathbf{y}) + \gamma(\mathbf{y} - \mathbf{q})$, where $\gamma \in \mathbb{R}$ is an arbitrary number, and $\mathbf{q} = \mathbb{E}[\tilde{\mathbf{Y}}]$ is the observed prior for the label. As the free term does not depend on the prediction $\hat{\mathbf{y}}$, it does not matter from an optimization perspective.

Pick-all-labels A loss function under the pick-all-labels reduction can be written in linear form as $\ell(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{j=1}^m y_j \ell_j(\hat{\mathbf{y}})$, so that its unbiased form is $\tilde{\ell}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{j=1}^m \frac{y_j}{p_j} \ell_j(\hat{\mathbf{y}})$.

For the generic case of non-decomposable multilabel losses, Theorem 19 provides the following expression

$$\tilde{\ell}(\hat{\mathbf{y}}, \tilde{\mathbf{y}}) = \left(\prod_{j: \tilde{y}_j=1} \frac{1}{p_j} \right) \sum_{\mathbf{y}' \leq \tilde{\mathbf{y}}} \ell(\hat{\mathbf{y}}, \mathbf{y}') \prod_{j': y_{j'}=1, y_{j'}=0} (1 - p_{j'}). \quad (3.20)$$

The “sum over all sub-labelings”-structure of the equation indicates that evaluating this loss function can be expensive. To be precise, it requires $2^{\|\tilde{\mathbf{y}}\|_1}$ evaluations of the original function. As XMC generally assumes $\|\mathbf{y}\|_1 \ll m$,⁴ however, this is still *much* more efficient than applying the result of Van Rooyen and Williamson [152] (c.f. section 3.1) directly, as this requires evaluating *all* 2^m possible values of $\ell(\hat{\mathbf{y}}, \cdot)$, which is intractable even for moderate values of m .

⁴see Table 1.1, column LpP, for the average number of labels per point in example datasets.

A second problem of (3.20) is that the division by products of propensities can lead to terms that are strongly amplified. Consequently, (3.20) is an estimator with very high variance.

Therefore, we want to find an upper-bound surrogate that trades off some bias against a reduction in variance. To handle both normalized one-vs-all and pick-all-labels reductions, the random variable $\mathbf{T} := \mathbf{Y}/\|\mathbf{Y}\|$, representing normalized labels, is introduced. Thus, these reductions can be written as

$$\ell(\hat{\mathbf{Y}}, \mathbf{Y}) = h(\hat{\mathbf{Y}}) + \sum_{j=1}^m T_j g_j(\hat{\mathbf{Y}}). \quad (3.21)$$

Under the convention $0/0 = 0$, the normalized label T_j can be written also as

$$T_j = \frac{Y_j}{1 + \sum_{j' \neq j} Y_{j'}}, \quad (3.22)$$

separating the contributions of noise in the numerator and denominator. This enables the application of Jensen's inequality to yield

$$\mathbb{E}[\tilde{T}_j] = \mathbb{E}\left[\frac{\tilde{Y}_j/p_j}{1 + \sum_{j' \neq j} \tilde{Y}_{j'}/p_{j'}}\right] \geq \mathbb{E}[T_j]. \quad (3.23)$$

If $g_j \geq 0$, which is true for pick-all-labels, but not for one-vs-all, then replacing \mathbf{T} by $\tilde{\mathbf{T}}$ gives an upper bound on the unbiased estimate.

The bias-variance trade-off can be demonstrated in two ways. First, using an artificially generated dataset (including its labels), the proposed unbiased estimator and upper bound can be used to estimate the instance-averaged recall, which has the form of a normalized pick-all-labels reduction. The resulting estimates, along with their standard deviation, are plotted in Figure 3.2.

The blue line shows that if propensities are sufficiently high, the unbiased estimator works well, and has reasonably low variance, but quickly deteriorates for propensities less than 50%. Both the upper bound and just applying the loss function without any special considerations for missing labels lead to stable variance, but the bias of the upper bound is generally larger than the vanilla estimate, which, at least in this case, consistently underestimates the true recall.

To analyze the effects of the proposed estimators also in a training setting, Publication II derives a version of the AmazonCat [105] dataset where only the 100 most frequent (and thus expected to be least likely to have low propensity) labels are kept. Two versions of the training split are created, one in which these 100 labels are kept for all instances, and a second in which an artificial propensity is imposed, removing the labels from some instances.

This gives four different ways of training a classifier: **a)** Using the original loss function on clean training data, **b)** using the original loss function on noisy training data, **c)** using the unbiased estimator, and **d)**, using the

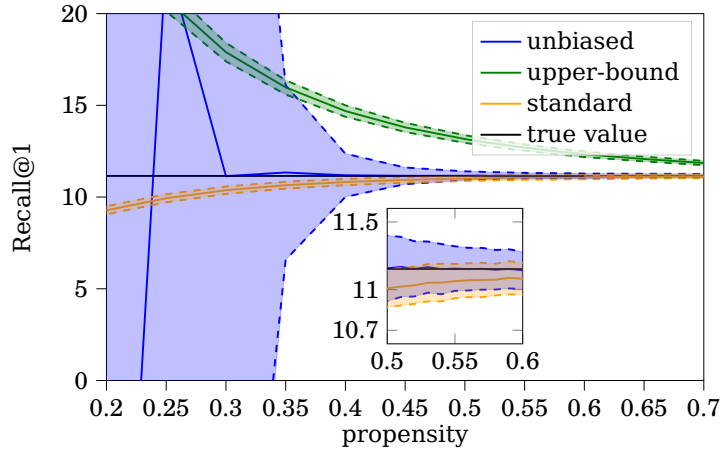


Figure 3.2. Unbiased estimate of per-example recall with artificial data (c.f. section 6 of Publication II). The shaded region corresponds to one standard deviation, estimated over 100 repetitions. The black line denotes the true recall. Adapted from Publication II, Figure 2.

upper bound. Additionally, there are three ways to evaluate the resulting classifiers: Using the original loss function on clean test data, using the original loss function on clean training data, and using the unbiased estimate on noisy training data.

We can then train multiple classifiers using varying amounts of regularization, to see how the bias-variance trade-off of the different training methods varies. For the normalized pick-all-labels reduction applied to the categorical cross-entropy loss, the results of such an experiment are presented in Figure 3.3.

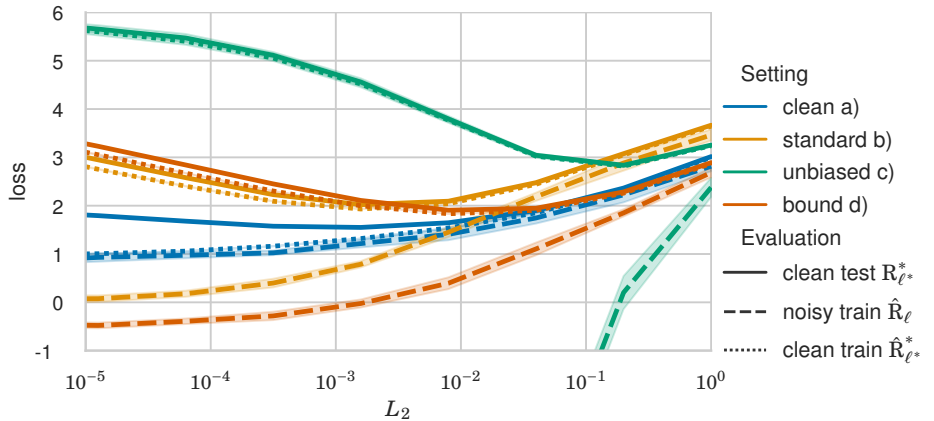


Figure 3.3. Normalized categorical cross-entropy for different regularization strengths, evaluated on noisy training data, clean training data, and clean test data. The gaps between dashed and dotted lines correspond to overfitting to the noise pattern, the smaller gaps between dotted and solid lines show the generalization gaps due to the finite training sample.

It becomes immediately obvious that the unbiased estimate is quite unusable in this setup, with its training error diverging to large negative numbers and high testing error except in cases with very strong regularization. The second main takeaway is that the phenomenon of overfitting is more nuanced in the missing-labels setup than in a standard machine learning setting: When training on clean data (blue lines), the estimated loss on the clean and noisy training set version (dotted and dashed lines) are similar, and there is a significant gap towards the loss on the test set. But in the other cases, we observe an overfitting to the noise pattern itself,⁵ with most of the gap being between the noise and clean training set.

For a more thorough discussion of this topic, as well as further experiments using other loss functions and reductions, refer to section 7 in Publication II.

3.4 Critical Discussion of the Propensity Model

The results presented above all assume access to a model for the propensity $p_j = \mathbb{P}[\tilde{Y}_j = 1 \mid Y_j = 1]$. In XMC, the de-facto standard for modelling the propensity is the formula developed by Jain et al. [64]. In their model, the propensity of a label can be determined as a function of its number of positives, and the total number of points in the dataset, through

$$p_j = p_{jpv}(n_j, n) := \frac{1}{1 + (\log n - 1)(b + 1)^a e^{-a \log(n_j + b)}} \quad (3.24)$$

Despite its widespread adoption, however, the model has some important flaws, as elucidated in Publication III:

Dataset generalization The two free parameters a and b have been estimated only for two datasets and show distinctly different values for these two datasets, $a \in \{0.5, 0.6\}$, $b \in \{0.4, 2.6\}$. For all other datasets, the common approach is to use the arithmetic mean of the two values, without any further justification.

Scaling behaviour If you consider gathering more data points for a dataset using the same generating distribution, that is, n and n_j increasing proportionally, the expression in (3.24) *always* converges to one.

Independence assumptions By construction, this model assumes that **a**) labels go missing independently of the instance’s features, **b**) the rate with which labels go missing only depends on the number of positives n_j , and **c**) whether one label goes missing has no influence on whether another label goes missing.

⁵Propensity-overfitting in counterfactual estimation [145]

Of course, in order to make any progress on the problem of missing labels, some assumptions have to be made, even if they are not perfectly realistic. Publication III provides two main lines of evidence that the assumptions are problematic. First, an attempt to reproduce the estimated propensities on the Wikipedia dataset, as shown in Figure 3.4, demonstrates that while the propensities, on average, might follow the curve in (3.24), but that there is a large amount of variance among labels of the same frequency. As such, modelling propensities as a function of label frequency is questionable.

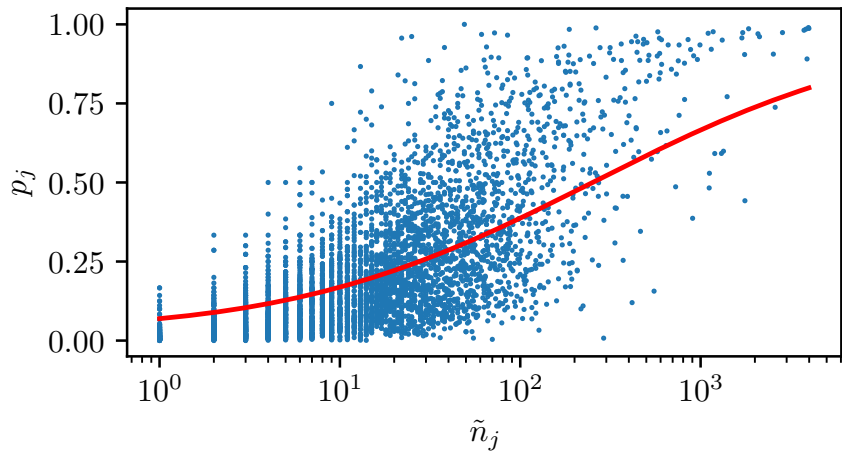


Figure 3.4. Reproduced propensities for individual labels based on the observed label frequency \tilde{n}_j , as well as the fit proposed by Jain et al. [64]. Figure reproduced from Publication III

Second, the resulting “unbiased” estimates are problematic in their own right. Evaluating the unbiased estimators for PfastreXm1 [64] gives peculiar results, as the calculated numbers according to (2.11) can exceed 100% by far. Yet in typical XMC publications, e.g., the results collecting in Bhatia et al. [11], values for PSP metrics are reported in the expected range of 0-100%.

This discrepancy arises because, following Jain et al. [64]:

Note that the gain G could be greater than 1 due to the propensities. Therefore, for greater interpretability, Table 3 reports $100 \cdot G(\{\hat{y}_i\})/G(\{y_i\})$ [...].

normalized versions (2.12) of the “unbiased estimator” are used, invalidating the interpretation of unbiased estimates, and hiding problems with model misspecification or excessive variance.

The differences can be tremendous, as illustrated in Table 3.1. As the unnormalized values are consistently larger than 100%, it seems unlikely that these extreme values are caused by high variance alone, indicating substantial model misspecification. This could be due to a direct mismatch

of propensity values, or due to a correlation between instances \mathbf{X} and masks \mathbf{M} , or a combination of the two.

Despite the interpretability problem of the numbers resulting from normalization, it is important to note that the scaling factor depends only on the dataset, and not on any predictions, so it does not influence relative comparisons of models on a fixed dataset. However, as shown with control experiments on synthetic data in Table 5 of Publication III, if the propensity model is mismatched with the data, then model selection can fail, and unnormalized estimates are a tool to detect this in some cases.

Table 3.1. Normalized and unnormalized propensity-scored precision of PfastreXML [64], when using the p_{JPV} model, with $a = 0.5$, $b = 0.4$ for WikiLSTHC-325K and $a = 0.6$, $b = 2.6$ for Amazon-670K. Reproduced from Publication III.

	WikiLSTHC-325K			Amazon-670K		
PSP(%)	@1	@3	@5	@1	@3	@5
Normalized	31.16	31.80	33.35	29.93	31.26	32.80
Unnormalized	196.96	118.54	85.28	326.47	282.28	250.57

Having established these deficiencies of the commonly used JPV propensity model, Publication III broadens its scope and looks at alternative ways to specify propensities. To that end, it investigates the Yahoo R3 dataset.⁶ A subset of the data was created by explicitly querying users about (instance, label) pairs, so that on this subset knowledge of the mask \mathbf{M} is available and the propensity on the rest of the dataset can be estimated directly by matching the label priors on the subsets.

Several alternative propensity models are evaluated on that data, including fitting a Richardson curve [131] and jointly learning the propensity with the rest of the model following Teisseyre et al. [148]. All three approaches (Richardson, joint learning, and JPV) led to improvements over the naïve baseline of training the model without any adjustments for missing labels, and perform worse than using the directly-estimated label-wise propensities. Notably, for this particular dataset, the JPV model was least successful.

Finally, it is important to note that in contemporary usage, normalized PSP metrics are treated primarily as an indicator for the performance of a model on the long tail of labels, disregarding their original derivation as unbiased estimates under a missing-label model. In that capacity, they are as principled as other ad-hoc weighting schemes that upweight infrequent labels.

⁶<https://webscope.sandbox.yahoo.com>

3.5 Summary and Discussion

With the results of Publications I to III in mind, we can revisit the questions posed in the introduction.

Unbiased estimates for general multilabel loss functions, RQ 2a Unbiased estimates can be formed for any multilabel loss function, under the assumption that labels go missing independently with known propensities. For decomposable loss function, equation (3.9) gives a simple way to calculate such estimates, whereas non-decomposable functions require the much more involved expression (3.20).

Efficiency of unbiased estimates, RQ 2b From a computational point of view, the decomposable unbiased estimates cause negligible overhead, but the compute cost for the non-decomposable estimator is exponential in the number of observed labels. While the average number of labels per instance in XMC settings is low enough to keep calculating this estimator feasible, a single outlier with a large number of labels can make the calculation intractable. Thus, one may require additional techniques, such as randomly subsampling the terms in (3.20), for such cases.

More problematic than the computational cost, though, is the statistical efficiency of these estimates. For estimating the performance of fixed models, this manifests in an increased variance of the estimator; for training, the unbiased loss versions provide a new source of overfitting that can overshadow the benefits of unbiased estimation by far. This is particularly problematic for non-decomposable losses, where the unbiased version contains products of propensities in the denominator, making them unsuitable as training objectives in XMC.

Surrogates for unbiased estimates, RQ 2c For (decomposable) OVA-based loss functions that are based on surrogates of the 0-1 loss, we can form surrogates of the unbiased version of the 0-1 loss to circumvent some of the optimization problems: The new loss function will be lower-bounded and convex, but there is still an increase in Lipschitz constant that implies that more samples are required. For (decomposable) PAL-based losses, the situation is even simpler: Their unbiased version is just a convex combination of the original multiclass loss, and no additional steps need to be taken to ensure lower-boundedness.

The situation is less favourable for normalized reductions. In the PAL-N case, we found an upper-bound that is far less susceptible to overfitting than the unbiased estimate, so that it provides a viable option as a training objective. However, its advantage over losses that ignore the missing-labels problem altogether seems quite minor.

And for normalize OVA decompositions, the same trick as for PAL-N does not yield an upper bound, so at this point we do not have such a bound.

Overall, the results in this chapter indicate that the combination of increased variance and the lack of a reliable propensity model cause unbiased estimators to be of only limited use as a tool to address missing labels in XMC problems. For decomposable loss functions, the variance can be kept in check, and Publication I shows that surrogates for unbiased estimators can lead to improved results using the standard PSP metrics but without any additional test set with controlled subsampling of labels, it is difficult to say how much of the improvements is really because missing labels are adequately addressed, and how much is simply the result of optimizing the metric that is used for evaluation more directly.

As the contemporary usage of PSP metrics is more motivated by their increased weight on tail labels than by any missing-labels interpretation, however, the provided loss functions still have tangible benefit in this regard. The next chapter will discuss alternative performance measures for tail labels that do not rely on dataset-dependent hyperparameters.

4. Macro-averaged Performance Metrics

To motivate the need for tail-label adapted performance metrics, consider the following experiment, performed in Publication VII: We take the AmazonCat [105] dataset and train two different models. The first uses the full available label information, whereas the second discards all labels that are not among the 1000 most frequent labels entirely, i.e., in that training run, only 1000 columns of the label matrix have nonzero entries. Both models are then evaluated on the *full* test set.

The results of this experiment are presented in Table 4.1, which shows that standard performance metrics, including the “tail-label adapted” propensity-scored precision (PSP), are quite stable against this training set perturbation, especially at low values of k . This suggests that good predictive performance on the long tail is not actually a necessary condition to achieving high scores in these metrics.

In contrast, macro-averaged metrics, as described in the rest of this chapter, can be very sensitive to long-tail performance, suffering a substantial decline if the classifier does not receive any tail labels in its training set.

Table 4.1. Performance measures (%) on AmazonCat-13k of a classifier trained on the full set of labels and a classifier trained with only 1k head labels. Adapted from Publication VII.

Metric	full labels			head labels		
	@1	@3	@5	@1 (diff.)	@3 (diff.)	@5 (diff.)
Precision	93.03	78.51	63.74	93.08 (+0.05%)	76.42 (-2.66%)	58.21 (-8.67%)
nDCG	93.03	87.25	85.35	93.08 (+0.05%)	85.75 (-1.71%)	80.91 (-5.19%)
PSP	49.76	62.63	70.35	49.07 (-1.39%)	57.71 (-7.84%)	57.41 (-18.40%)
Macro-P	13.28	32.65	44.16	4.31 (-67.54%)	5.28 (-83.82%)	4.32 (-90.21%)
Macro-R	1.38	11.06	30.57	0.47 (-65.61%)	2.69 (-75.71%)	4.10 (-86.59%)
Macro-F1	2.26	14.67	32.84	0.74 (-67.37%)	3.10 (-78.88%)	3.77 (-88.51%)
Coverage	15.19	40.53	60.88	5.11 (-66.32%)	7.37 (-81.82%)	7.52 (-87.65%)

4.1 Background: Generalized performance metrics

In equation (2.1), the risk is written as an expectation over individual instances; it decomposes. However, it is also possible to define loss functions which need to combine information aggregated over multiple instances. Examples for such metrics, called *non-decomposable* or *generalized* performance measures, include the area under the ROC curve [33], geometric mean [33, 158, 108], as well as the F-measure [151, 93] as the harmonic mean of precision and recall.

Metric	Ψ	Metric	Ψ
Accuracy	$tp + tn$	Balanced Acc.	$\frac{tp}{2(tp+fn)} + \frac{tn}{2(tn+fp)}$
Precision	$\frac{tp}{tp+fp}$	Recall	$\frac{tp}{tp+fn}$
F_1	$\frac{2tp}{2tp+fn+fp}$	G-mean	$\sqrt{\frac{tp}{tp+fp} \cdot \frac{tp}{tp+fn}}$

Table 4.2. Examples of generalized performance measures, written in terms of confusion-matrix entries. Except for accuracy, which is a linear function, these metrics are non-decomposable.

Already in the binary case, non-decomposability significantly increases the complexity of even just defining which task precisely one wants to solve. One view, coming from an idealized, mathematical perspective, is to consider the limit of receiving an infinite amount of testing data. To evaluate the quality of a predictor in this setup, we need to evaluate a functional $\Psi[\hat{Y}, Y]$.¹ As both Y and \hat{Y} are binary variables, their joint distribution is fully characterized by the (population) confusion matrix²

$$C[\hat{Y}, Y] = \begin{pmatrix} \mathbb{E}[(1-Y)(1-\hat{Y})] & \mathbb{E}[Y(1-\hat{Y})] \\ \mathbb{E}[(1-Y)\hat{Y}] & \mathbb{E}[Y\hat{Y}] \end{pmatrix} \quad (4.1)$$

Thus, the goal is to optimize

$$Y^* = \underset{\hat{Y}}{\operatorname{argmax}} \Psi(C[\hat{Y}, Y]), \quad (4.2)$$

where Ψ is a confusion-matrix measure, and the optimal classifier is thresholding operation, with potential randomization at the threshold [140]. As the optimization is performed over the entire, infinite population of possible samples, this approach is called *population utility* (PU).

Equation (4.2) looks quite different from the definition of Bayes-optimal predictions given in (2.3), which considers the best prediction for each given input instance. Of course, for non-decomposable metrics, it is not possible to make predictions that are optimal at the level of a single instance,

¹We use square brackets to indicate that this is a functional, that is, a function the takes as inputs the random variables themselves, instead of acting on individual samples.

²Also called a *contingency table*

but we can define optimal decisions for any fixed, given set of instances. Collecting all instances into a matrix \mathbf{X} , and corresponding labels³ $\mathbf{Y} \in \{0, 1\}^n$ and predictions $\hat{\mathbf{Y}} \in \{0, 1\}^n$, we can consider a function $\Psi: \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{R}$ to evaluate all our predictions jointly. The corresponding version of (2.3), requiring us to be optimal in expectation over all possible ground-truth labelings that could be sampled for the given set of instances, is

$$\mathbf{y}^*(\mathbf{X}) = \underset{\hat{\mathbf{y}}}{\operatorname{argmax}} \mathbb{E}[\Psi(\hat{\mathbf{y}}, \mathbf{Y}) | \mathbf{X} = \mathbf{X}]. \quad (4.3)$$

As the optimization is over the expected performance on a fixed test set, this framework is called *expected test utility* (ETU).

Alternative names for PU are *expected utility optimization* and for ETU *decision-theoretic approach* [169]. Under mild assumptions, the ETU-optimal solution will converge to the PU-optimal solution as the test set size grows to infinity [30].

Methods exist for handling generalized performance measures in binary [17, 169, 67, 85, 30], multiclass [120, 121], and multilabel [29, 86, 83, 157] setups.

4.2 Macro-averaged performance metrics

Instead of just presenting a guess at a useful formula for a performance metric, let us try to take a more principled approach and derive a class of metrics by requiring certain properties.

Imagine you are supposed to compare different XMC models by mapping them to a single, scalar score. How can that be achieved? The typical approach is to collect a test set of n examples not yet seen during training. Then each model can make its predictions $\hat{\mathbf{Y}} \in \{0, 1\}^{n \times m}$ and compare them against the ground truth $\mathbf{Y} \in \{0, 1\}^{n \times m}$ though

$$\Psi: \{0, 1\}^{n \times m} \times \{0, 1\}^{n \times m} \rightarrow \mathbb{R}. \quad (4.4)$$

As the ordering of the instances in the training set is arbitrary, it should not have any influence on the model evaluation procedure. Similar, the mapping of labels to columns in the predicted/ground-truth matrix is immaterial to the success of the classifier. Thus, for arbitrary permutations matrices $\sigma \in S(n)$ and $\rho \in S(m)$, we have

$$\Psi(\hat{\mathbf{Y}}, \mathbf{Y}) = \Psi(\sigma \hat{\mathbf{Y}} \rho^T, \sigma \mathbf{Y} \rho^T). \quad (4.5)$$

This can be extended to support, e.g., weighted labels and instances by allowing for additional per-label/per-instance metadata that gets permuted

³Note that we are still in the binary setting, hence we have one binary value for each of the n instances, instead of one value for each class label m .

alongside the matrices. This metadata should be fixed a-priori, i.e., it cannot depend on any prediction.

Additionally, the function Ψ should be compatible with the following partial order introduced on the predicted labels \hat{Y} for each Y :

$$\hat{Y} \leq_Y \hat{Y}' \Leftrightarrow \forall i, j : \mathbb{1}[\hat{Y}_{ij} = Y_{ij}] \leq \mathbb{1}[\hat{Y}'_{ij} = Y_{ij}] ; \quad (4.6)$$

that is, \hat{Y} contains a mistake *at least* at all the positions where \hat{Y}' contains a mistake. As such, \hat{Y}' is strictly no worse than \hat{Y} , and we want that reflected in our valuation,

$$\hat{Y} \leq_Y \hat{Y}' \implies \Psi(\hat{Y}, Y) \leq \Psi(\hat{Y}', Y). \quad (4.7)$$

As our ultimate goal for this chapter is to find both useful metrics *and* an efficient way to make optimal predictions for that metric, we impose an additional constraint: we want the metric to be decomposable.

The metrics used in the preceding chapter, and typically reported in XMC [11], all have in common that they are decomposable over instances, that is, they can be computed at the level of an individual instance. The overall performance of the method is evaluated based on the average over all instances according to (2.1). For the purpose of finding tail-label oriented metrics, we take the orthogonal approach, assuming that the metric can be decomposed over labels instead, that is, we have

$$\Psi(\hat{Y}, Y) = \sum_{j=1}^m \psi(\hat{\mathbf{y}}_{:j}, \mathbf{y}_{:j}). \quad (4.8)$$

Metrics of this form are called *macro-averaged* metrics, and form a subset of generalized performance measures. This can be generalized slightly by allowing the functions ψ to be different for each label.⁴

$$\Psi(\hat{Y}, Y) = \sum_{j=1}^m \psi_j(\hat{\mathbf{y}}_{:j}, \mathbf{y}_{:j}). \quad (4.9)$$

This form does not yet guarantee that Ψ is invariant under instance re-ordering, which can be ensured by requiring the ψ_j to be functions of the confusion matrix of $\hat{\mathbf{y}}_{:j}$ and $\mathbf{y}_{:j}$. Thus, we can equivalently write

$$\Psi(\hat{Y}, Y) = \sum_{j=1}^m \psi_j(\hat{C}(\hat{\mathbf{y}}_{:j}, \mathbf{y}_{:j})), \quad (4.10)$$

where \hat{C} denotes the mapping of binary predictions and labels to the empirical confusion matrix,

$$\hat{C}(\hat{\mathbf{y}}, \mathbf{y}) := \sum_{i=1}^n \begin{pmatrix} (1 - \hat{y}_i)(1 - y_i) & \hat{y}_i(1 - y_i) \\ (1 - \hat{y}_i)y_i & \hat{y}_i y_i \end{pmatrix}. \quad (4.11)$$

⁴The main purpose of this generalization is to allow the PSP metric and other forms of label-weighted precision to be included in this framework.

In fact, any function that decomposes linearly over the labels and is invariant to the ordering of instances can be written in the form (4.9), as shown in Appendix A1 of Publication VII.

The equation (4.10) calculates a performance metric for any given set of labels and predictions, i.e., it acts in the ETU setting. By replacing the empirical confusion matrix \hat{C} with the population confusion matrix C ,

$$C[\hat{Y}, Y] := \begin{pmatrix} \mathbb{E}[(1 - \hat{Y})(1 - Y)] & \mathbb{E}[\hat{Y}(1 - Y)] \\ \mathbb{E}[(1 - \hat{Y})Y] & \mathbb{E}[\hat{Y}Y] \end{pmatrix}, \quad (4.12)$$

we can instead define a macro-averaged population utility as

$$\Psi(\hat{Y}, Y) = \sum_{j=1}^m \psi_j(C[\hat{Y}_{:j}, Y_{:j}]). \quad (4.13)$$

At first glance, (4.10) and (4.13) look like they would be straightforward to solve. By construction, the objective separates into completely independent problems for each label, so optimizing each binary problem individually would yield the optimal solution to the multilabel problem, and standard techniques for binary non-decomposable metrics could be applied [86]. However, prediction in XMC is typically performed “at k ”, that is, for each presented test instance, the algorithm is required to return exactly k candidate labels, that is, $\|\hat{y}\|_0 = \|\hat{y}\|_1 = k$. This constraint couples the binary problems, as the labels now have to compete for the k available slots.

4.3 Optimal predictions under the ETU setting

In the ETU setting, we are given a fixed sample of test instances X , for which we need to predict the unknown labels to maximize

$$\begin{aligned} Y^* &= \max_{\hat{Y} \in \{0,1\}^{n \times m}} \mathbb{E} \left[\sum_{j=1}^m \psi_j(\hat{C}(\hat{y}_{:j}, y_{:j})) \mid \mathbf{X} = X \right]. \\ &\text{s.t. } \|\mathbf{y}_i\|_1 = l \quad \forall i \in [n] \end{aligned} \quad (4.14)$$

One can show (Appendix A.2 in Publication VII) that this objective can be written as a function of the predictions and the marginal probability for each label at a given instance, $\boldsymbol{\eta}_j(x) = \mathbb{E}[Y_j \mid \mathbf{X} = x]$. Crucially, this means that a standard learning algorithm, e.g., based on the OVA decomposition, can be used to learn an approximation $\boldsymbol{\eta}$ efficiently, and we “only” have to find an inference algorithm based on predicted probabilities.

Publication VII tackles this problem by making what we term a *semi-empirical* approximation: We can parametrize the confusion matrix metric by true positives t , ground-truth positives p and predicted positives q . Then we define semi-empirical quantities

$$\tilde{t} := \mathbb{E}[t \mid \mathbf{X} = X], \tilde{p} := \mathbb{E}[p \mid \mathbf{X} = X], \tilde{q} := \mathbb{E}[q \mid \mathbf{X} = X] = q, \quad (4.15)$$

where we use that the predicted positives are a deterministic function of the predicted labels. Our approximation then replaces

$$\Psi_{\text{ETU}} := \mathbb{E}[\phi(t, q, p) \mid \mathbf{X} = \mathbf{X}] \approx \phi(\tilde{t}, q, \tilde{p}) =: \tilde{\Psi}_{\text{ETU}}, \quad (4.16)$$

that is, we exchange expectation and ϕ . If ϕ is linear in t and p (but not necessarily in q), this substitution is exact, otherwise it incurs an error bounded by $\mathcal{O}(\sqrt{n})$ according to Theorem 5.2 in Publication VII if ϕ fulfills specific Lipschitzness conditions.

If the function ϕ is *linear* in t and q (but not necessarily in p , which is constant for a fixed test set),

$$\phi(t, q, p) = f_t(p) \cdot t + f_q(p) \cdot q + f_p(p), \quad (4.17)$$

the entire objective $\tilde{\Psi}_{\text{ETU}}$ decomposes linearly over instances as well as labels. This implies that the optimization can be performed for each instance independently, where the optimal prediction resolves to taking a weighted top- k of the label probabilities, with the weighting factors, henceforth called *gains* g , determined by the coefficients of the linear function:

$$g_j(x) := \eta_j(x) f_t(p_j) + f_q(p_j). \quad (4.18)$$

The more interesting case is if the function does not decompose. In this general case, we can make progress as follows: If we assume that we already know the optimal predictions for all but one instance in the test set, the prediction problem for the single remaining instance turns into a linear optimization problem whose solution is again a weighted top- k . This insight allows us to construct a block-coordinate-ascent (BCA) algorithm, where each block of coordinates is the predictions for a single instance. By iterating over the entire dataset in randomized order and updating predictions for each instance to be optimal conditioned on the others, we can refine the prediction matrix until we achieve a stationary point in which single-instance changes no longer lead to any improvement.

In order for this approach to be efficient, the individual steps need to be performant. Assuming we have a model that provides $\boldsymbol{\eta}$ in $\mathcal{O}(m)$, we can select the weighted top- k in $\mathcal{O}(m \log(k))$ time. However, à priori, the weights depend on the confusion matrix, which requires $\mathcal{O}(m \cdot n)$ operations to be calculated. Fortunately, we can keep track of the confusion matrix in an online fashion: When updating the prediction for an instance, we subtract the contribution made by the old predictions, and add the contribution by the new predictions. This is possible because calculating the confusion matrix is linear over the instances. Therefore, the BCA step can be implemented in $\mathcal{O}(m)$.

4.4 Optimal predictions under the PU setting

Now, we turn to the problem of optimizing macro averages in the PU setting. In this case, the target to optimize is

$$\mathbf{h}^* = \max_{\mathbf{h} \in \mathcal{H}} \sum_{j=1}^m \psi_j(C[h_j(\mathbf{X}), \mathbf{Y}_j]), \quad (4.19)$$

s.t. $\|\mathbf{h}(\mathbf{x})\|_1 = k \quad \forall \mathbf{x} \in \mathcal{X}$

where $\mathcal{H}: \mathcal{X} \rightarrow [0, 1]^m$ represents the class of probabilistic multilabel classifiers.⁵

To develop an algorithm for optimal inference, let us first start again with the simpler case of *linear* metrics. Let $C = C[\hat{\mathbf{Y}}, \mathbf{Y}] := [C[\hat{Y}_1, Y_1], \dots, C[\hat{Y}_m, Y_m]]$ be the confusion tensor of the multilabel problem, then we consider metrics of the form

$$\Psi(\hat{\mathbf{Y}}, \mathbf{Y}) = \langle G, C[\hat{\mathbf{Y}}, \mathbf{Y}] \rangle, \quad (4.20)$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product when interpreting these tensors as vectors, i.e., we calculate a weighted sum of all the entries of the confusion tensor.

We can show (Publication VIII, Theorem 4.1) that in this case, the optimal solution is given by a top-k selection of an *affine* transformation of label probabilities

$$\mathbf{h}^*(\mathbf{x}) = \text{top}_k(\mathbf{a} \cdot \boldsymbol{\eta}(\mathbf{x}) + \mathbf{b}), \quad (4.21)$$

where

$$a_j = G_{00}^j + G_{11}^j - G_{01}^j - G_{10}^j, b_j = G_{01}^j - G_{00}^j. \quad (4.22)$$

This result can be directly used to tackle the non-linear case. Under some regularity assumptions, the solution of the nonlinear optimization problem (4.19) can be found as the solution to a specific linear optimization problem,

$$\mathbf{h}^* = \underset{\mathbf{h}}{\text{argmax}} \langle G, C[\mathbf{h}] \rangle, \quad (4.23)$$

where the gain tensor G given through $G = \nabla \Psi(C[\mathbf{h}^*])$ (Publication VIII, Theorem 4.2).

While this does not help directly - in order to know the coefficients G for the linear problem, the optimal confusion tensor $C(\mathbf{h}^*)$ already needs to be known - it points the way toward an algorithmic solution. For a given candidate solution \mathbf{h}' , we can calculate the corresponding confusion tensor C' . If \mathbf{h}' were optimal, then by the theorem solving the resulting linear problem with $G = \nabla \Psi(C[\mathbf{h}^*])$. If C' is not optimal, we still can solve the linear problem, and get a new candidate \mathbf{h}'' . Because Ψ is not linear, this new solution need not be better when evaluated with Ψ instead of G ; however, we can do a line search interpolating between \mathbf{h}' and \mathbf{h}'' . The update step is illustrated in Figure 4.1.

⁵Any vector $\hat{\mathbf{h}} \in [0, 1]^m$ with $\|\hat{\mathbf{h}}\|_1 = k$ can be turned into a probabilistic prediction $\hat{\mathbf{H}} \in [0, 1]^m$ with $\|\hat{\mathbf{H}}\|_1 = k$ almost surely by using, e.g., Madow's sampling[103].

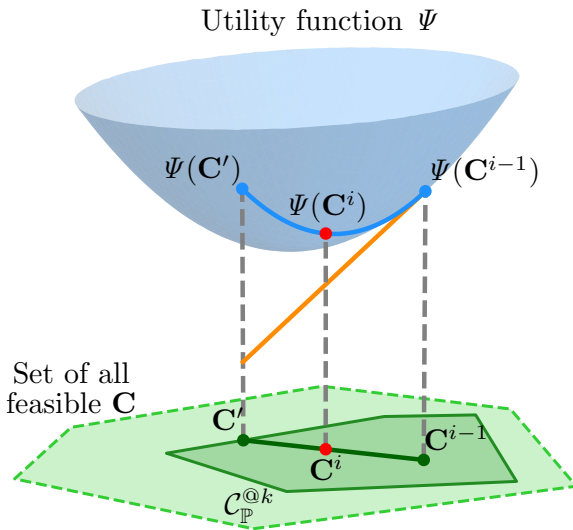


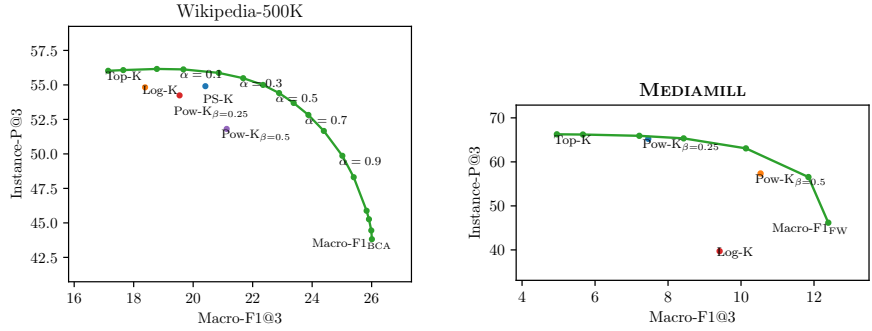
Figure 4.1. Frank-Wolfe optimization step: A linearization of the metric Ψ at the current confusion matrix C is generated, and the optimal solution in the linearization is calculated. The next classifier (and correspondingly, the next confusion matrix) is then generated by a line search between the old and the linearized solution. Figure from the poster accompanying Publication VIII, by Marek Wydmuch.

This is the Frank-Wolfe [44] algorithm for constrained optimization. If Ψ is concave and sufficiently smooth, this algorithm is guaranteed to converge to an optimal solution with high probability, as shown in Publication VIII, Theorem 5.1. This theorem is an adaptation of a similar statement given in Narasimhan et al. [120], which in turn is based on the analysis of Frank-Wolfe by Jaggi [63].

4.5 Results and discussion

While the block-coordinate and Frank-Wolfe algorithm are very effective of optimizing their target metric, as shown in the respective papers, this usually coincides with a drastic decrease in traditional performance metrics. Fortunately, we can also consider the precision-at- k metric in the macro-average framework: precision-at- k simply arises when the binary metric to average over is chosen to be the true positive rate. Since any linear interpolation of two macro-averaged metrics is again a macro-averaged metric, this means that we can smoothly trade-off precision-at- k with a secondary, tail-label adapted metric.

Because we chose a setup in which the prediction problem is separated into two phases, learning a model $\hat{\eta}$ to predict marginal label probabilities,



(a) Block-coordinate-ascent on the Wiki500 dataset.

(b) Frank-Wolfe on Mediamill.

Figure 4.2. Interpolation between precision-at-3 and macro-averaged F1-measure-at-3, as well as power-law (POW) and logarithmic (LOG) label weightings.

and then running a comparatively cheap inference procedure (neither BCA nor FW take more than a few minutes) on top, we can efficiently generate optimal predictions for many different performance metrics.

In Figure 4.2, we present an interpolation between macro-F1 and precision metrics, as well as some other empirical weighting schemes based on either a power-law or a logarithm of the label frequencies. The graph shows that initially, the macro-F1 (and thus performance on tail labels) can be improved substantially without affecting the precision much, but at some point, improvements in F1 can only be achieved by massive decrease of precision. Still, our proposed method appears to be Pareto-optimal compared with the other heuristics that we compared against in this example.

Interestingly, and contrary to the ETU setting, none of the arguments required for deriving the optimal PU predictions need to have Ψ decomposable over labels; any concave metric that is a function of the confusion tensor can be optimized with the FW algorithm. This difference arises because in the PU case, we optimize over a convex set (of probabilistic classifiers), whereas the optimization target in ETU is discrete. In particular, the inner optimization step, finding optimal predictions for a single instance given predictions for the rest of the dataset, becomes tractable for metrics that decompose over labels.

Above, the algorithms were presented based on access to the full set of predicted probabilities $\hat{\eta}$, yet for efficiency reasons, many XMC methods chose to employ some form of shortlisting, and thus only predict a small subset of the entries of $\hat{\eta}$. Publications VII and VIII show that the algorithm remains viable if one uses shortlisted predictions, implicitly assuming any non-shortlisted value to be zero.

In the context of the questions raised in the introduction of the thesis, we can draw the following conclusions:

Long-tail performance metrics, RQ 1a We have shown that the class of macro-averaged performance metrics contains metrics that are sensitive to tail-label performance. However, not every macro-average is, e.g., precision-at- k also falls within this framework. Additionally, from a practical perspective, the most interesting metrics are probably an interpolation between traditional P@k and a tail-adapted macro-averages. While this introduces a new hyperparameter, the interpolation factor, plots like Figure 4.2 are easily produced and can help make an informed choice.

Optimal predictions, RQ 1b In the PU case, for concave and sufficiently smooth objectives, the Frank-Wolfe algorithm converges to the optimal solution. For the ETU case, making optimal predictions would require solving an intractable combinatorial optimization problem.

Efficient approximately-optimal predictions, RQ 1c In order to make the ETU task tractable, we apply a semi-empirical approximation, and derive a block-coordinate-ascent algorithm that produces locally-optimal predictions. The cost of both the BCA and FW algorithms grows linearly in the label size, and both algorithms can be combined with typical XMC shortlisting approaches for further efficiency gains. Making optimal predictions for different objectives can be done cheaply, without requiring expensive retraining of the XMC model that just provides probability estimates $\hat{\eta}$.

5. Computational Efficiency

Whereas the preceding two chapters focused on problems related to the data distribution in XMC, this section deals with the computational challenges that arise from the large label space. First, we will demonstrate that even simple and well-established baselines can benefit from a more careful implementation on modern hardware, and some simple tricks exploiting the highly imbalanced nature of the XMC problem. Then we specifically turn to the problem of memory cost, which is generally less well explored than the computational aspects of XMC.

5.1 Efficient linear models for imbalanced multilabel problems

Dismec [4], at its core, is highly-parallelized training of independent linear classifiers for each label. In principle, this embarrassingly parallel task should scale perfectly to large multicore processors. In practice, however, speedup quickly saturates as core counts are increased. This is because of the memory bottleneck: Even though all calculations are independent and run on separate CPU cores, they have to share the same memory bandwidth, which quickly saturates. This is particularly problematic on large NUMA system, where RAM is partitioned in multiple NUMA domains, such as CSC's Mahti, see Figure 5.1.

The feature matrix X that needs to be repeatedly read by all binary sub-problems will reside in only one NUMA node, effectively swamping this part of memory, while leaving the other nodes underutilized. Therefore, the first optimization Publication IV makes is to generate copies of the feature matrix in each NUMA node, pin individual threads to cores, and then ensure that each thread accesses the copy that is closest to its core.

After fixing this parallelization problem, taking a closer look at a profile of where most of the time is spent reveals a peculiar phenomenon: The first few iterations of the conjugate-gradient based optimizer [77, 47] are taking *much* longer time than later iterations, despite seemingly doing the same operations, at least when the equations are written down in matrix

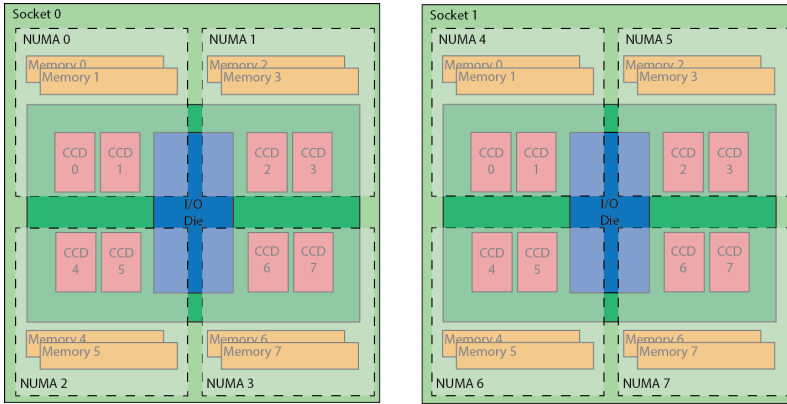


Figure 5.1. Memory arrangement in a single compute node on the CSC Mahti cluster. A node contains two CPUs with four NUMA nodes each. When reading memory connected to one socket from the CPU on the other, data needs to be transferred through an interconnect that can easily become a bottleneck. Image reproduced from <https://docs.csc.fi/computing/systems-mahti/>.

form. The observed speedup is due to the fact that the conjugate-gradient optimizer performs many Hessian-vector products in its inner loop. As Dismec uses a squared hinge loss, instances that are classified correctly with sufficient margin do not contribute to the Hessian, and thus may be skipped in the computations [42, 77]:

$$\mathbf{H}\mathbf{d} = \sum_{i=1}^n \mathbf{x}_i \phi''(y_i \mathbf{w}^\top \mathbf{x}_i) \langle \mathbf{x}_i, \mathbf{d} \rangle + \mathbf{d} = \sum_{i \in \mathcal{A}} \phi''(\mathbf{x}_i^\top \mathbf{w} \cdot y_i) y_i \langle \mathbf{x}_i, \mathbf{d} \rangle \cdot \mathbf{x}_i + \mathbf{d}, \quad (5.1)$$

where \mathcal{A} denotes the set of active instances, i.e., those for which classification has not yet reached a sufficient margin. As the weight vector approaches its optimum, fewer and fewer instances need to be taken into account; the time per iteration decreases.

Thus, the main idea of Publication IV is to find an initial vector that can be calculated quickly, and that leads to a very sparse Hessian matrix from the beginning. Initial progress had been made on that problem by Fang et al. [43], but further improvements remained possible. Due to the high imbalance of most binary problems comprising the XMC task, a simple idea would be to predict all-negative, e.g., by setting a negative bias term at initialization. However, as investigated in our paper, such a scheme does not lead to good speed-ups, as the sparsity in the Hessian does not persist into the second iteration.

Instead, we chose an initial weight vector that generates a separating hyperplane between the centers of masses of the positive and the negative instances. This can be calculated efficiently by precomputing the sum over all instances once, and then adjusting for the sum over all positives within each binary subproblem. The choice of such an initial weight is illustrated in Figure 5.2.

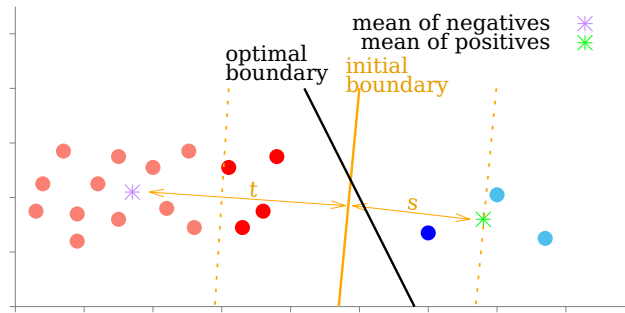


Figure 5.2. Mean-separating initialization. The initial weight vector (solid yellow line) is chosen such that it separates the mean feature of the positive instances (green) from the mean of the negatives (violet). Any instance that is more than the margin-width from the plane thusly defined (i.e., outside the area demarcated by the dashed yellow lines) does not contribute to the Hessian, so the computations need only involve a small subset of instances. As we expect far more negative than positive instances, the distance of the hyperplane from the positive center s is smaller than that from the negative center t . Figure from Publication IV.

Despite the simplicity of this approach, we find speed-ups between $1.8\times$ and $5\times$ compared to the default of initializing with zero.

5.2 Background: Memory efficiency and sparsity

Memory Efficiency for XMC While techniques such as probabilistic labels trees and shortlisting are great at reducing the runtime cost of extreme classifiers, they do address the enormous memory cost associated with the classification layer. The cost is twofold: For one, the activation memory associated with calculating and storing all logits, loss values, and derivatives is enormous, but can be mostly avoided by clever implementation [66]. Second, the weights of the classifier itself consume a huge amount of memory; during training, this may be compounded by the fact that typical deep learning optimizers store additional buffers for momentum terms, and single-precision master copies in the case of mixed-precision training.

One way to reduce the memory consumption of the XMC model is through weight pruning, a technique that has long been employed to reduce inference costs [90, 39, 53, 116, 56, 92, 160]. In the XMC setting, the Dismec model trained on tf-idf Amazon-670k consumes 3.75 GB of memory, only a tiny fraction of the memory cost of a dense matrix of size $4B \times 670091 \times 135909 \approx 364\text{GB}$. In the case of linear models specifically, pruning also helps in reducing memory during training, as weights can be pruned as soon as individual binary problems are solved. For deep learning approaches, however, all labels are trained simultaneously, and thus post-training pruning does not help with training memory.

An alternative method that promises true memory reductions is Mach [106]: Here, instead of learning one weight vector per label, the labels are randomly hashed, and prediction is performed on the much more coarse-grained label hashes (meta labels). In order to be able to make predictions at the level of single labels, multiple independently hashed classifiers are trained. It can be shown that, in a multiclass setting, the label probabilities can be reconstructed from the predicted probabilities of meta-labels if sufficiently many hash functions are used. Overall, this results in a memory cost that grows only logarithmic in the number of labels. Unfortunately, the efficiency comes at the cost of predictive performance, where typical OVA-based models outperform Mach [11].

The idea explored in this section is to achieve dynamic sparse training, that is, to use sparse classifier weights throughout the training process. This could be seen as moving the pruning step from inference to training time, or as a modification of Mach, which could be seen as a very specific sparse training setup in which the sparsity is fixed in both connectivity and weights.

Dynamic Sparse Training Before presenting the application of dynamic sparse training (DST) to extreme classification, this subsection will give a brief overview over the field.

Sparsity has a long history in the training of neural networks, going back to at least the 80s and 90s [90, 55, 143]. In addition to reduced computational and memory cost, sparsity also promises potential improvements in generalization, e.g., iterative pruning and retraining techniques based on the “lottery ticket hypothesis” [45]. A recent survey on sparsity in neural networks is Hoefler et al. [59].

DST, in particular, refers to the application of sparsity throughout the training process, instead of removing weights from a densely-trained network. In order to determine the optimal connectivity, DST algorithms generally follow a prune-rewire-regrow cycle: First, suitable connections are identified for pruning and removed. Then, new connections are generated according to some criterion. Finally, regular gradient-descent based training is performed to update the weights, until the next pruning step starts.

A simple way to implement this is to prune weights by magnitude (Set, [115]) or whenever they change their sign (DeepR, [8]), and rewire by uniformly random choice. More sophisticated methods utilize gradient information [41], dense momentum buffers [31], or second-order information [90].

Sparsity and Hardware Unstructured removal of weights, while very appealing from a theoretical perspective, leads to much less tangible improvement than one would naïvely expect when actually executed on modern hardware. Issues arise because of the non-uniform memory access [112] required, work imbalances [167] between different neurons, and the unavailability of tensor cores for unstructured sparsity [180]. This is exacerbated by the

fact that sparse matrices also need to store their location of non-zeros in some form, so that actual memory improvements are less than the sparsity ratio.

While sophisticated implementations can mitigate some of the impact of these problems [159, 46, 124], practical speedups significantly lag behind the theoretical improvements that would be expected based on the amount of floating-point operations that need to be executed. Structured sparsity, on the other hand, is much more amenable to hardware: In the most extreme case, entire neurons or even layers are removed [161, 116, 94, 56], allowing computations to retain their dense structure. However, such crude model surgery often comes at the cost of diminished accuracy. Less drastic are block-sparse [50, 119] formats, in which fixed-shape sub-blocks of weight matrices are retained or removed.

Even more fine-grained are so-called “ $n:m$ ” sparsities: Within each section of m possible weights, at most n can have a non-zero value [167]. This makes the sparsity pattern more uniform, streamlining memory access and work balancing. In particular, for $n = 2$ and $m = 4$, this can be implemented directly at the hardware level, and is supported in NVidia GPUs starting from the Ampere generation.¹ In addition to reductions in quality at low values of m , for the purpose of training a network, $n:m$ sparsity also suffers from the fact that in the backward pass, the transpose of the weight matrix is needed, which does not necessarily have the same $n:m$ pattern, unless transposable masks are explicitly enforced [62].

5.3 DST for learning the classification layer

As a first step towards employing sparsity for the XMC problem, Publication V considers the reduced problem of training just the classification layer using dynamic sparsity. In particular, input features are pre-generated either using generic fastText [75, 14] embeddings, or by dropping the classification layer from a trained CascadeXML [79] network and extracting the penultimate embeddings.

First attempts of using pre-existing implementations of sparsity in the tensorflow [1] framework failed due to the representation of sparse matrices in tensorflow: They are represented in *coordinate* (COO) format, that is, each nonzero is stored as one floating-point value and two integer indices indicating row and column. In tensorflow, these indices use 64-bit integers, thus massively increasing the memory footprint of the sparse representation.

Therefore, we opt for a custom implementation of a slightly structured sparsity pattern that constrains the number of non-zero elements to be

¹www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf

identical in every row (neuron). In addition to the computational benefit, such a structure also ensures that no label can be pruned away entirely. A visualization of different sparse storage formats is shown in Figure 5.3.

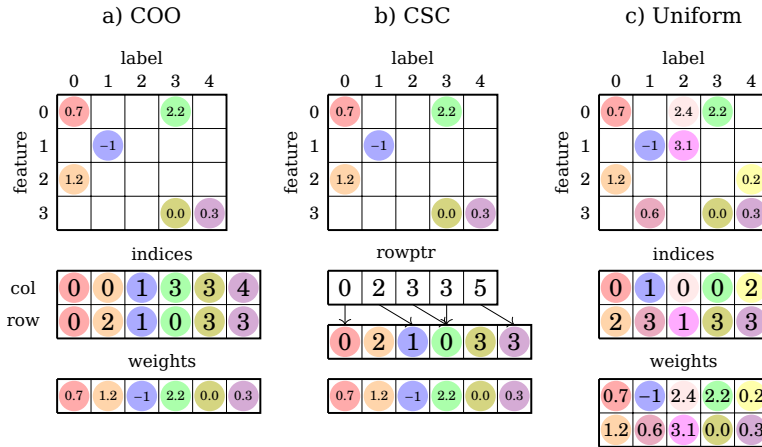


Figure 5.3. Schematic depiction of different sparse matrix formats. Note that in CoO format a), the indices array in Algorithm 1 is of shape $2 \times \text{nnz}$. In uniform format c) it is $\text{nnz per column} \times \text{labels}$, and hence only half as big, compared to the CoO format, for the same number of nonzeros. Figure adapted from Publication V.

We implemented custom sparse kernels that used constant fan-in sparsity so that a single 16-bit integer index is enough to identify the location of each nonzero. In order to enable an efficient implementation, we store input features in column-major order, contrary to the default storage format in tensorflow and pytorch [125]. Column-major ordering ensure that the same features of different input instances are next to each other in memory, making the memory access pattern of the matrix multiplication more regular.

Algorithm 1 Calculation of the score for a single label and instance for fixed fan-in sparsity Figure 5.3 with s non-zeros for each label.

```

value = 0;
for i in range(s):
    source = indices[i, label]
    feature = features[batch, source]
    value += feature * weights[i, label]
data[batch, label] = value

```

While the constant fan-in format works well for the forward pass, during gradient computation, one needs to multiply with the *transpose* of the weight matrix, which does not benefit from the fan-in constraint. By using, as in Publication IV, a squared hinge loss, we can exploit the same implicit negative mining effect that also enabled much of the efficiency of Dismec++: After the first few optimization steps, a large fraction of the logit gradient

will be zero, because the corresponding labels are easy negatives. By exploiting this gradient sparsity, we can also bring down the computational cost of the backward pass.

A fast and memory-efficient implementation in itself is not useful unless the resulting method remains close in predictive performance to the original dense baseline, and in particular, outperforms a smaller dense model of equal memory consumption. Simply training a fixed fan-in sparse classification layer using SET did not, however, achieve this, and instead resulted in severe underfitting.

To compensate, an intermittent layer of intermediate size can be added. Even if this layer has much more neurons than the embedding layer, the overall increase in parameters is only moderate, as the parameter count for the sparse classification layer is independent of the size of the preceding layer. For e embedding dimensions, h intermediate neurons, a fan-in of f and m labels, the total parameter count is given by $e \cdot h + f \cdot m$. In Publication V, we used fan-ins $f \in \{32, 64\}$, and intermediate layers between 16k and 100k units.

Not unexpectedly, increasing the number of parameters, either through the intermediate layer or through the fan-in, leads to improved classification performance, but of course also increase memory consumption. Interestingly, we found that when training based on generic features (Slice’s fastText [14, 75]) features, the sparse network (with increased representation capacity due to the intermediate layer) actually results in better test-set performance than the dense baseline, but when using highly-adapted features, extracted from fully-trained CascadeXML [79], we were not able to match the baseline. Looking at the performance on the training set, we find that for both types of features, the sparse model achieves slightly less precision.

With our implementation, the time *per epoch* for the dense model and the sparse model are approximately equal, but the sparse model requires more epochs to be trained [96].

5.4 End-to-end DST training

Given that above, we found that training just the extreme classifier suffers from underfitting, one might assume that end-to-end training the embedding model together with the classifier allows for the embeddings to be co-adapted to the sparse classifier. This is the aim of Publication VI, which puts a sparse classification layer on top of a Bert [32]-based embedding model.

However, initial experiments showed disappointing results, with dynamic sparse training lagging far behind the dense baseline. In particular, end-to-end training seemed to achieve precision much lower than adapting just the

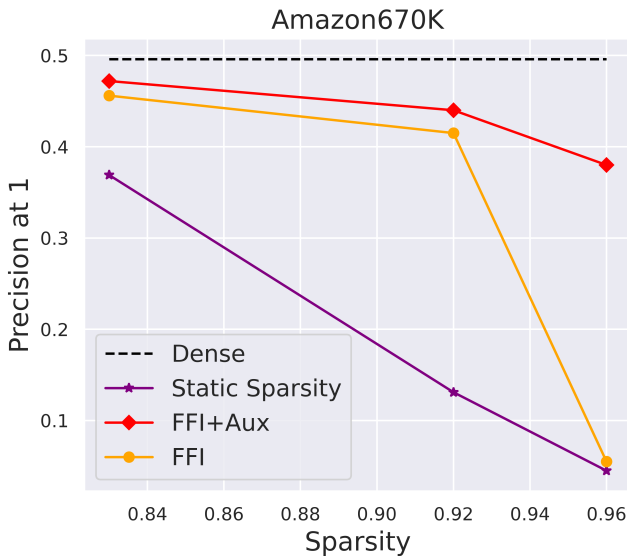


Figure 5.4. At low sparsities, even static sparsity leads to reasonable performance. As the sparsity increases, static sparsity becomes useless and dynamic sparse training is needed. At the highest level of sparsity, the auxiliary loss becomes critical for preserving accuracy. Figure adapted from Publication VI.

sparse classifier as in Publication V. A sparse classification layer impedes learning of the *dense* embedding model!

Somehow, the gradients backpropagated from the sparse layer do not provide a good signal to fine-tune the Bert model. To mitigate this, we add an auxiliary loss that bypasses the extreme layer. As auxiliary loss, we chose a meta-label classification task, that is, we add a second classification head whose targets are a coarse-grained version of the original label vector, i.e., we use the same targets as other XMC methods use for the purpose of label shortlisting.

As the auxiliary loss ultimately would lead to different optimal weights than the actual task loss, we only enable it for the initial epochs of training and turn it off later. The end-to-end training with auxiliary loss obviates the need for the intermediate layer introduced in the preceding section.

Figure 5.4 shows how increasing sparsity requires more sophisticated training methods to limit the impact on performance: For 66% sparsity, even a static selection of the nonzero location performs reasonably well; at 95% sparsity, the auxiliary loss is essential to prevent the model from becoming completely useless.

5.5 Discussion and Outlook

In this chapter, computational aspects of the XMC problem have been investigated. In Publication IV, it was demonstrated that even algorithms as simple as linear models can benefit greatly from hardware and data-distribution aware implementations. Despite the impressive speedup that Dismec++ achieves over Dismec, we believe that there is still much further potential. One such example is sorting the (sparse) input features according to their frequency to improve cache hits rates [102]. As such, Publication IV should be considered more as a first, albeit substantial, step towards truly hardware-efficient linear XMC methods as requested in RQ 3.

While an important (and still surprisingly effective) baseline, linear models have been surpassed by transformer-based models, which come with a much-increased demand for memory during training. Publications V and VI show that dynamic sparse training allows for much reduced memory cost at only moderate impact to the quality of the final model. As such, we do not quite match the predictive performance of the dense baseline, and while we can reduce the running time of a single epoch, the requirement for more training epochs negates any benefit in total running time, so we can consider RQ 4 as only partially solved. The main argument in favour of sparsity is the tremendous reduction in memory requirements.

An alternative to sparsity that enjoys much better hardware acceleration would be to use low-bit datatypes, like 16 and even 8-bit floating point numbers. This avenue is being investigated in Zhang et al. [176], which at the time of this writing is still under review, and show promising results that combine memory reduction and walltime speedups.

6. Summary and Conclusion

This thesis investigated multilabel problems with very large label spaces (extreme classification) from the perspectives of missing labels, long tails, and computational efficiency.

The method of unbiased estimators was applied to address the missing-labels problem, and such estimators derived for a large class of loss functions and evaluation metrics typical of extreme classification. Unfortunately, the unbiased estimators may be non-lower-bounded and consequently difficult to optimize for without severe overfitting. Therefore, surrogates are proposed that are more benign to optimization. A significant flaw that remains, though, is the reliance propensity estimates, which we showed to be rather problematic in their form currently established in extreme classification. Future research thus needs to find alternative ways to treat missing labels, e.g., data-cleaning approaches, or produce improved propensity models.

To handle the long-tailed nature of the label distribution characteristic of extreme classification, macro-averaged metrics are proposed. As predictions are typically performed with a budget, i.e., exactly k labels are to be selected, the optimal predictions for different labels are coupled through this constraint, and existing prediction algorithms cannot be applied directly. The thesis presents two new algorithms that handle the constrained prediction in two different statistical frameworks: making optimal predictions on an infinite population, and making optimal predictions on a given test set. In practice, using a purely tail-label oriented measure will lead to many irrelevant predictions; however, the framework allows for easy interpolation between traditional precision-at- k metrics and tail-oriented macro-averages, and we find that it is possible to improve tail-label performance significantly without much effect to the traditional metrics.

Towards improved computational efficiency, the thesis makes two contributions. First, an improved implementation of large-scale linear classifiers adapted to modern many-core systems is presented, and enhanced by a new weight initialization scheme that allows skipping the most expensive iterations of the second-order optimizer, leading to significant speed-ups.

Second, extreme classification is combined with dynamic sparse training in order to train an extreme classification model in a way in which weight matrices are sparse throughout the entire training process. This reduces the memory requirements and makes it possible to train with a much larger label space on any given GPU. Unfortunately, sparsity is ill-suited to utilize the full computing power of a GPU. Consequently, despite doing much fewer computations, the sparse implementation does not lead to faster trainings. Combining sparsity with existing approaches for label shortlisting, which eliminate the extreme classification layer as the key bottleneck in terms of computation, is an exciting avenue for future research, offering the possibility of combining the memory savings of the method presented here with fast training speeds.

References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [2] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24, 2013.
- [3] Mario Almagro, Raquel Martínez Unanue, Víctor Fresno, and Soto Montalvo. Icd-10 coding of spanish electronic discharge summaries: An extreme classification problem. *IEEE Access*, 8:100073–100083, 2020.
- [4] Rohit Babbar and Bernhard Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 721–729, 2017.
- [5] Rohit Babbar and Bernhard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8):1329–1351, 2019.
- [6] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.
- [7] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.
- [8] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.
- [9] Samy Bengio, Krzysztof Dembczynski, Thorsten Joachims, Marius Kloft, and Manik Varma. Extreme Classification (Dagstuhl Seminar 18291). Technical report, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. URL <https://drops.dagstuhl.de/entities/document/10.4230/DagRep.8.7.62>.

- [10] Alina Beygelzimer, John Langford, Yuri Lifshits, Gregory Sorkin, and Alex Strehl. Conditional probability tree estimation analysis and algorithms. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 51–58, Arlington, Virginia, USA, 2009. AUAI Press. ISBN 9780974903958.
- [11] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016. URL <http://manikvarma.org/downloads/XC/XMLRepository.html>.
- [12] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, 28, 2015.
- [13] Alberto Blanco, Alicia Perez, and Arantza Casillas. Extreme multi-label icd classification: sensitivity to hospital service and time. *IEEE Access*, 8: 183534–183545, 2020.
- [14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.
- [15] Anirudh Buvanesh, Rahul Chand, Jatin Prakash, Bhawna Paliwal, Mudit Dhawan, Neelabh Madan, Deepesh Hada, Vidit Jain, Sonu Mehta, Yashoteja Prabhu, et al. Enhancing tail performance in extreme classifiers by label variance reduction. In *The Twelfth International Conference on Learning Representations*, 2023.
- [16] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019.
- [17] Kian Ming Adam Chai. Expectation of f-measures: Tractable exact computation and some empirical observations of its properties. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 593–594, 2005.
- [18] Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. Large-scale multi-label text classification on EU legislation. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6314–6322, Florence, Italy, July 2019. Association for Computational Linguistics.
- [19] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3163–3171, New York, NY, USA, 2020. Association for Computing Machinery.
- [20] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [21] Beidi Chen, Tharun Medini, James Farwell, Charlie Tai, Anshumali Shrivastava, et al. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems*, 2:291–306, 2020.
- [22] Si-An Chen, Jie-Jyun Liu, Tsung-Han Yang, Hsuan-Tien Lin, and Chih-Jen Lin. Even the simplest baseline needs careful re-investigation: A case study on xml-cnn. In *Proceedings of the 2022 Conference of the North American*

- Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1987–2000, 2022.
- [23] Eli Chien, Jiong Zhang, Cho-Jui Hsieh, Jyun-Yu Jiang, Wei-Cheng Chang, Olgica Milenkovic, and Hsiang-Fu Yu. Pina: leveraging side information in extreme multi-label classification via predicted instance neighborhood aggregation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 5616–5630, 2023.
- [24] Yu-Ting Chou, Gang Niu, Hsuan-Tien Lin, and Masashi Sugiyama. Unbiased risk estimators can mislead: A case study of learning with complementary labels. In *International conference on machine learning*, pages 1929–1938. PMLR, 2020.
- [25] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.
- [26] Kunal Dahiya, Ananye Agarwal, Deepak Saini, K Gururaj, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International conference on machine learning*, pages 2330–2340. PMLR, 2021.
- [27] Kunal Dahiya, Deepak Saini, Anshul Mittal, Ankush Shaw, Kushal Dave, Akshay Soni, Himanshu Jain, Sumeet Agarwal, and Manik Varma. 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*, pages 31–39, 2021.
- [28] Kunal Dahiya, Nilesh Gupta, Deepak Saini, Akshay Soni, Yajun Wang, Kushal Dave, Jian Jiao, Gururaj K, Prasenjit Dey, Amit Singh, et al. 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, 2023.
- [29] Krzysztof Dembczynski, Arkadiusz Jachnik, Wojciech Kotłowski, Willem Waegeman, and Eyke Hüllermeier. Optimizing the f-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *International conference on machine learning*, pages 1130–1138. PMLR, 2013.
- [30] Krzysztof Dembczyński, Wojciech Kotłowski, Oluwasanmi Koyejo, and Nagarajan Natarajan. Consistency analysis for binary classification revisited. In *International Conference on Machine Learning*, pages 961–969. PMLR, 2017.
- [31] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.
- [32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.
- [33] Chris Drummond and Robert C Holte. Severe class imbalance: Why better algorithms aren't the answer. In *European Conference on Machine Learning*, pages 539–546. Springer, 2005.
- [34] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International conference on machine learning*, pages 1386–1394. PMLR, 2015.

- [35] Marthinus C Du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. *Advances in neural information processing systems*, 27, 2014.
- [36] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [37] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11*, page 1097-1104, Madison, WI, USA, 2011. Omnipress.
- [38] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213-220, 2008.
- [39] Andries P Engelbrecht and Ian Cloete. A sensitivity analysis algorithm for pruning feedforward neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 2, pages 1274-1278. IEEE, 1996.
- [40] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18-36, 2004.
- [41] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, pages 2943-2952. PMLR, 2020.
- [42] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871-1874, 2008.
- [43] Huang Fang, Minhao Cheng, Cho-Jui Hsieh, and Michael Friedlander. Fast training for large-scale one-versus-all linear classifiers using tree-structured initialization. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 280-288. SIAM, 2019.
- [44] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95-110, 1956.
- [45] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.
- [46] Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1-14. IEEE, 2020.
- [47] Leonardo Galli and Chih-Jen Lin. A study on truncated newton methods for linear classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [48] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93-107, 2015.
- [49] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie, editors, *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 518-529. Morgan Kaufmann, 1999. ISBN 1-55860-615-7.

- [50] Scott Gray, Alec Radford, and Diederik P Kingma. Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224*, 3(2):2, 2017.
- [51] Chuan Guo, Ali Mousavi, Xiang Wu, Daniel N Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. Breaking the glass ceiling for embedding-based classifiers for large output spaces. *Advances in Neural Information Processing Systems*, 32, 2019.
- [52] Nilesh Gupta, Sakina Bohra, Yashoteja Prabhu, Saurabh Purohit, and Manik Varma. Generalized zero-shot extreme multi-label learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 527–535, 2021.
- [53] Masafumi Hagiwara. Removal of hidden units and weights for back propagation networks. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 1, pages 351–354. IEEE, 1993.
- [54] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems*, 31, 2018.
- [55] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.
- [56] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.
- [57] Djoerd Hiemstra. A probabilistic justification for using tf \times idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3:131–139, 2000.
- [58] Clifford Hildreth. A quadratic programming procedure. *Naval research logistics quarterly*, 4(1):79–85, 1957.
- [59] Torsten Hoefer, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22 (241):1–124, 2021.
- [60] Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, and Yanning Chen. Liger kernel: Efficient triton kernels for llm training. *arXiv preprint arXiv:2410.10989*, 2024. URL <https://arxiv.org/abs/2410.10989>.
- [61] Chao-Wei Huang, Shang-Chi Tsai, and Yun-Nung Chen. PLM-ICD: Automatic ICD coding with pretrained language models. In Tristan Naumann, Steven Bethard, Kirk Roberts, and Anna Rumshisky, editors, *Proceedings of the 4th Clinical Natural Language Processing Workshop*, pages 10–20, Seattle, WA, July 2022. Association for Computational Linguistics.
- [62] Itay Hubara, Brian Chmiel, Moshe Isard, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34:21099–21111, 2021.
- [63] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pages 427–435. PMLR, 2013.

- [64] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 935–944, 2016.
- [65] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 528–536, 2019.
- [66] Vidit Jain, Jatin Prakash, Deepak Saini, Jian Jiao, Ramachandran Ramjee, and Manik Varma. Renee: End-to-end training of extreme classification models. *Proceedings of Machine Learning and Systems*, 5, 2023.
- [67] Martin Jansche. A maximum expected utility framework for binary sequence labeling. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 736–743, 2007.
- [68] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4): 422–446, 2002.
- [69] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. Extreme f-measure maximization using sparse probability estimates. In *International conference on machine learning*, pages 1435–1444. PMLR, 2016.
- [70] Kalina Jasinska-Kobus, Marek Wydmuch, Krzysztof Dembczynski, Mikhail Kuznetsov, and Robert Busa-Fekete. Probabilistic label trees for extreme multi-label classification. *arXiv preprint arXiv:2009.11218*, 2020.
- [71] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR, 2018.
- [72] Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):7987–7994, May 2021.
- [73] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 781–789, 2017.
- [74] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.
- [75] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.
- [76] Guik Jung, Junghoon Shin, and Sangjun Lee. Impact of preprocessing and word embedding on extreme multi-label patent classification tasks. *Applied Intelligence*, 53(4):4047–4062, 2023.

- [77] S Sathiya Keerthi, Dennis DeCoste, and Thorsten Joachims. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6(3), 2005.
- [78] Sujay Khandagale, Han Xiao, and Rohit Babbar. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109(11): 2099-2119, 2020.
- [79] Siddhant Kharbanda, Atmadeep Banerjee, Erik Schultheis, and Rohit Babbar. Cascadexml: Rethinking transformers for end-to-end multi-resolution training in extreme multi-label classification. *Advances in neural information processing systems*, 35:2074-2087, 2022.
- [80] Siddhant Kharbanda, Atmadeep Banerjee, Devaansh Gupta, Akash Palrecha, and Rohit Babbar. 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*, pages 760-769, 2023.
- [81] Siddhant Kharbanda, Devaansh Gupta, Erik Schultheis, Atmadeep Banerjee, Cho-Jui Hsieh, and Rohit Babbar. Gandalf: Learning label-label correlations in extreme multi-label classification via label features. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1360-1371, 2024.
- [82] Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30, 2017.
- [83] Wojciech Kotlowski and Krzysztof Dembczyński. Surrogate regret bounds for generalized classification performance metrics. In *Asian Conference on Machine Learning*, pages 301-316. PMLR, 2016.
- [84] Wojciech Kotlowski, Marek Wydmuch, Erik Schultheis, Rohit Babbar, and Krzysztof Dembczynski. A general online algorithm for optimizing complex performance metrics. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 25396-25425. PMLR, 21-27 Jul 2024.
- [85] Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent binary classification with generalized performance metrics. *Advances in neural information processing systems*, 27, 2014.
- [86] Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent multilabel classification. *Advances in Neural Information Processing Systems*, 28, 2015.
- [87] Abhishek Kumar, Shankar Vembu, Aditya Krishna Menon, and Charles Elkan. Beam search algorithms for multilabel learning. *Machine learning*, 92:65-89, 2013.
- [88] Leah S Larkey and W Bruce Croft. Combining classifiers in text categorization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 289-297, 1996.
- [89] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188-1196. PMLR, 2014.

- [90] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.
- [91] Joo Hyung Lee, Wonpyo Park, Nicole Elyse Mitchell, Jonathan Pilault, Johan Samir Obando Ceron, Han-Byul Kim, Namhoon Lee, Elias Frantar, Yun Long, Amir Yazdanbakhsh, et al. Jaxpruner: A concise library for sparsity research. In *Conference on Parsimony and Learning*, pages 515–528. PMLR, 2024.
- [92] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=B1VZqjAcYX>.
- [93] David D Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254, 1995.
- [94] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=rJqFGTslg>.
- [95] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124, 2017.
- [96] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR, 2021.
- [97] Yang Liu, Hua Cheng, Russell Klopfer, Matthew R Gormley, and Thomas Schaaf. Effective convolutional attention network for multi-label clinical document classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5941–5953, 2021.
- [98] Zichang Liu, Zhaozhuo Xu, Alan Ji, Junyan Zhang, Jonathan Li, Beidi Chen, and Anshumali Shrivastava. Halos: Hashing large output space for cheap inference. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 110–125, 2022.
- [99] Philip M Long and Rocco A Servedio. Random classification noise defeats all convex potential boosters. In *Proceedings of the 25th international conference on Machine learning*, pages 608–615, 2008.
- [100] Romain Lopez, Inderjit S Dhillon, and Michael I Jordan. Learning from extreme bandit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8732–8740, 2021.
- [101] Eneldo Loza Mencía and Johannes Fürnkranz. *Efficient multilabel classification algorithms for large-scale problems in the legal domain*. Springer, 2010.
- [102] Wayne Luk, Ce Guo, Henning Funke, Jens Teubner, Erik Schultheis, Rohit Babbar, Helena Kotthaus, and Peter Marwedel. Hardware-aware execution. In Katharina Morik and Peter Marwedel, editors, *Volume 1 Machine Learning under Resource Constraints - Fundamentals*, pages 249–304. De Gruyter, Berlin, Boston, 2023.
- [103] William G Madow. On the theory of systematic sampling, ii. *The Annals of Mathematical Statistics*, 20(3):333–354, 1949.

- [104] Laxmaiah Manchikanti, Alan D Kaye, Vijay Singh, and Mark V Boswell. The tragedy of the implementation of icd-10-cm as icd-10: Is the cart before the horse or is there a tragic paradox of misinformation and ignorance? *Pain physician*, 18(4):E485, 2015.
- [105] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.
- [106] Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. *Advances in Neural Information Processing Systems*, 32, 2019.
- [107] Sonu Mehta, Jayashree Mohan, Nagarajan Natarajan, Ramachandran Ramjee, and Manik Varma. Astra: Accurate and scalable anns-based training of extreme classifiers. *arXiv preprint arXiv:2409.20156*, 2024.
- [108] Aditya Menon, Harikrishna Narasimhan, Shivani Agarwal, and Sanjay Chawla. On the statistical consistency of algorithms for binary classification under class imbalance. In *International Conference on Machine Learning*, pages 603–611. PMLR, 2013.
- [109] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International conference on machine learning*, pages 125–134. PMLR, 2015.
- [110] Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Multilabel reductions: what is my loss optimising? *Advances in Neural Information Processing Systems*, 32, 2019.
- [111] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2021.
- [112] Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.
- [113] Anshul Mittal, Kunal Dahiya, Sheshansh Agrawal, Deepak Saini, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Decaf: Deep extreme classification with label features. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 49–57, 2021.
- [114] Anshul Mittal, Kunal Dahiya, Shreya Malani, Janani Ramaswamy, Seba Kuruvilla, Jitendra Ajmera, Keng-hao Chang, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Multi-modal extreme classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12393–12402, 2022.
- [115] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.
- [116] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, 2016.
- [117] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *International workshop on artificial intelligence and statistics*, pages 246–252. PMLR, 2005.

- [118] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, 2018.
- [119] Sharan Narang, Eric Undersander, and Gregory Diamos. Block-sparse recurrent neural networks. *arXiv preprint arXiv:1711.02782*, 2017.
- [120] Harikrishna Narasimhan, Harish Ramaswamy, Aadirupa Saha, and Shivani Agarwal. Consistent multiclass algorithms for complex performance measures. In *International Conference on Machine Learning*, pages 2398–2407. PMLR, 2015.
- [121] Harikrishna Narasimhan, Harish G Ramaswamy, Shiv Kumar Tavker, Drona Khurana, Praneeth Netrapalli, and Shivani Agarwal. Consistent multiclass algorithms for complex metrics and constraints. *Journal of Machine Learning Research*, 25(367):1–81, 2024.
- [122] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.
- [123] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Cost-sensitive learning with noisy labels. *The Journal of Machine Learning Research*, 18(1):5666–5698, 2017. Publisher: JMLR. org.
- [124] Mahdi Nikdan, Tommaso Pegolotti, Eugenia Iofinova, Eldar Kurtic, and Dan Alistarh. Sparseprop: Efficient sparse backpropagation for faster training of neural networks at the edge. In *International Conference on Machine Learning*, pages 26215–26227. PMLR, 2023.
- [125] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [126] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.
- [127] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, pages 993–1002, 2018.
- [128] Yashoteja Prabhu, Aditya Kusupati, Nilesh Gupta, and Manik Varma. Extreme regression for dynamic search advertising. In *Proceedings of the 13th international conference on web search and data mining*, pages 456–464, 2020.
- [129] Mohammadreza Qaraei and Rohit Babbar. Meta-classifier free negative sampling for extreme multilabel classification. *Machine Learning*, 113(2): 675–697, 2024.
- [130] Mohammadreza Mohammadnia Qaraei, Sujay Khandagale, and Rohit Babbar. Why state-of-the-art deep learning barely works as good as a linear classifier in extreme multi-label text classification. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 223–228. i6doc. com, 2020.

- [131] Francis J Richards. A flexible growth function for empirical use. *Journal of experimental Botany*, 10(2):290–301, 1959.
- [132] Morgane Rivi re, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, L onard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ram , Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozinska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshov, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucinska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sj sund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.
- [133] TYLPG Ross and GKHP Doll r. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988, 2017.
- [134] Noveen Sachdeva, Lequn Wang, Dawen Liang, Nathan Kallus, and Julian McAuley. Off-policy evaluation for large action spaces via policy convolution. In *Proceedings of the ACM on Web Conference 2024*, pages 3576–3585, 2024.
- [135] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care ii: a public-access intensive care unit database. *Critical care medicine*, 39(5):952–960, 2011.
- [136] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. Galaxc: Graph neural networks with labelwise attention for extreme classification. In *Proceedings of the Web Conference 2021*, pages 3733–3744, 2021.
- [137] Yuta Saito and Thorsten Joachims. Off-policy evaluation for large action spaces via embeddings. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 19089–19122. PMLR, 17–23 Jul 2022.
- [138] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [139] Rajat Sen, Alexander Rakhlin, Lexing Ying, Rahul Kidambi, Dean Foster, Daniel N Hill, and Inderjit S Dhillon. Top-k extreme contextual bandits with

- arm hierarchy. In *International Conference on Machine Learning*, pages 9422–9433. PMLR, 2021.
- [140] Shashank Singh and Justin T Khim. Optimal binary classification beyond accuracy. *Advances in Neural Information Processing Systems*, 35:18226–18240, 2022.
- [141] Dezhao Song, Andrew Vold, Kanika Madan, and Frank Schilder. Multi-label legal document classification: A deep learning-based approach with label-attention and domain-specific pre-training. *Information Systems*, 106: 101718, 2022.
- [142] Alex Strehl, John Langford, Lihong Li, and Sham M Kakade. Learning from logged implicit exploration data. *Advances in neural information processing systems*, 23, 2010.
- [143] Nikko Ström. Sparse connection and pruning in large dynamic artificial neural networks. In *Fifth European Conference on Speech Communication and Technology*. Citeseer, 1997.
- [144] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 814–823, Lille, France, 07–09 Jul 2015. PMLR.
- [145] Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems*, 28, 2015.
- [146] Yukihiro Tagami. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 455–464, 2017.
- [147] Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *Advances in Neural Information Processing Systems*, 37:114147–114179, 2025.
- [148] Paweł Teisseyre, Jan Mielniczuk, and Małgorzata Łazęcka. Different strategies of fitting logistic regression for positive and unlabelled data. In *International Conference on Computational Science*, pages 3–17. Springer, 2020.
- [149] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [150] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109:475–494, 2001.
- [151] Cornelis Joost Van Rijsbergen. Foundation of evaluation. *Journal of documentation*, 30(4):365–373, 1974.
- [152] Brendan Van Rooyen and Robert C Williamson. A theory of learning with corrupted labels. *Journal of Machine Learning Research*, 18(228):1–50, 2018.

- [153] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. *Advances in neural information processing systems*, 28, 2015.
- [154] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [155] Sudheendra Vijayanarasimhan, Jonathon Shlens, Rajat Monga, and Jay Yagnik. Deep networks with large output spaces. *arXiv preprint arXiv:1412.7479*, 2014.
- [156] Thanh Vu, Dat Quoc Nguyen, and Anthony Nguyen. A label attention model for icd coding from clinical text. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3335–3341. International Joint Conferences on Artificial Intelligence Organization, 7 2020.
- [157] Willem Waegeman, Krzysztof Dembczyński, Arkadiusz Jachnik, Weiwei Cheng, and Eyke Hüllermeier. On the bayes-optimality of f-measure maximizers. *Journal of Machine Learning Research*, pages 3513–3568, 2014.
- [158] Shuo Wang and Xin Yao. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1119–1130, 2012.
- [159] Ziheng Wang. Sparsert: Accelerating unstructured sparsity on gpus for deep learning inference. In *Proceedings of the ACM international conference on parallel architectures and compilation techniques*, pages 31–42, 2020.
- [160] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, 2020.
- [161] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.
- [162] Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. A no-regret generalization of hierarchical softmax to extreme multi-label classification. *Advances in neural information processing systems*, 31, 2018.
- [163] Marek Wydmuch, Kalina Jasinska-Kobus, Rohit Babbar, and Krzysztof Dembczynski. Propensity-scored probabilistic label trees. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2252–2256, 2021.
- [164] Xiaobo Xia, Bo Han, Yibing Zhan, Jun Yu, Mingming Gong, Chen Gong, and Tongliang Liu. Combating noisy labels with sample selection by mining high-discrepancy examples. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1843, 2023.
- [165] Yuanhao Xiong, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. Extreme zero-shot learning for extreme text classification. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5455–5468, 2022.

- [166] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [167] Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. Balanced sparsity for efficient dnn inference on gpu. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5676–5683, 2019.
- [168] Hui Ye, Zhiyu Chen, Da-Han Wang, and Brian Davison. Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10809–10819. PMLR, 13–18 Jul 2020.
- [169] Nan Ye, Kian Ming Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measures: A tale of two approaches. In *Proceedings of the 29th International Conference on Machine Learning*, pages 289–296. Omnipress, 2012.
- [170] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553, 2017.
- [171] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multi-class and multilabel classification. In *International conference on machine learning*, pages 3069–3077. PMLR, 2016.
- [172] Ronghui You, Zihan Zhang, Ziyue Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in neural information processing systems*, 32, 2019.
- [173] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601. PMLR, 2014.
- [174] Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23(98):1–32, 2022.
- [175] Jinbin Zhang, Nasib Ullah, and Rohit Babbar. Zero-shot learning over large output spaces: Utilizing indirect knowledge extraction from large language models. *arXiv preprint arXiv:2406.09288*, 2024.
- [176] Jinbin Zhang, Nasib Ullah, Erik Schultheis, and Rohit Babbar. Elmo: Efficiency via low-precision and peak memory optimization in large output spaces. In *under submission*, 2025.
- [177] Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34:7267–7280, 2021.
- [178] Tianyi Zhang, Zhaozhuo Xu, Tharun Medini, and Anshumali Shrivastava. Structural contrastive representation learning for zero-shot multi-label text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4937–4947, 2022.

- [179] Zachariah Zhang, Jingshu Liu, and Narges Razavian. BERT-XML: Large scale automated ICD coding using BERT pretraining. In Anna Rumshisky, Kirk Roberts, Steven Bethard, and Tristan Naumann, editors, *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 24–34. Association for Computational Linguistics, November 2020.
- [180] Maohua Zhu, Tao Zhang, Zhenyu Gu, and Yuan Xie. Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 359–371, 2019.

Business, Economy
Art, Design, Architecture
Science, Technology
Crossover
| Doctoral Theses

Aalto DT 180/2025

ISBN 978-952-64-2731-7
ISBN 978-952-64-2730-0 (pdf)

Aalto University
School of Science
Department of Computer Science
aalto.fi