Taming Pretrained Transformers for Extreme Multi-label Text Classification

Wei-Cheng Chang
Carnegie Mellon University

Hsiang-Fu Yu
Amazon

Kai Zhong
Amazon

Yiming Yang
Carnegie Mellon University

Inderjit S. Dhillon
Amazon & UT Austin

ABSTRACT

We consider the extreme multi-label text classification (XMC) problem: given an input text, return the most relevant labels from a large label collection. For example, the input text could be a product description on Amazon.com and the labels could be product categories. XMC is an important yet challenging problem in the NLP community. Recently, deep pretrained transformer models have achieved state-of-the-art performance on many NLP tasks including sentence classification, albeit with small label sets. However, naively applying deep transformer models to the XMC problem leads to sub-optimal performance due to the large output space and the label sparsity issue. In this paper, we propose X-Transformer, the first scalable approach to fine-tuning deep transformer models for the XMC problem. The proposed method achieves new state-of-the-art results on four XMC benchmark datasets. In particular, on a Wiki dataset with around 0.5 million labels, the prec@1 of X-Transformer is 77.28%, a substantial improvement over state-of-the-art XMC approaches Parabel (linear) and AttentionXML (neural), which achieve 68.70% and 76.95% precision@1, respectively. We further apply X-Transformer to a product2query dataset from Amazon and gained 10.7% relative improvement on prec@1 over Parabel.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; *Natural language processing*; • **Information systems** → *Information retrieval.*

KEYWORDS

Transformer models, eXtreme Multi-label text classification

ACM Reference Format:
Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. 2020. Taming Pretrained Transformers for Extreme Multi-label Text Classification. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403368

1 INTRODUCTION

We are interested in the Extreme multi-label text classification (XMC) problem: given an input text instance, return the most relevant labels from an enormous label collection, where the number of labels could be in the millions or more. One can view the XMC problem as learning a score function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, that maps an (instance, label) pair $(\mathbf{x}, \mathbf{y})$ to a score $f(\mathbf{x}, \mathbf{y})$. The function f should be optimized such that highly relevant $(\mathbf{x}, \mathbf{y})$ pairs have high scores, whereas the irrelevant pairs have low scores. Many real-world applications are in this form. For example, in E-commerce dynamic search advertising, $\mathbf{x}$ represents an item and $\mathbf{y}$ represents a bid query on the market [20, 21]. In open-domain question answering, $\mathbf{x}$ represents a question and $\mathbf{y}$ represents an evidence passage containing the answer [4, 11]. In the PASCAL Large-Scale Hierarchical Text Classification (LSHTC) challenge, $\mathbf{x}$ represents an article and $\mathbf{y}$ represents a category of the Wikipedia hierarchical taxonomy [17].

XMC is essentially a text classification problem on an industrial scale, which is one of the most important and fundamental topics in machine learning and natural language processing (NLP) communities. Recently, deep pretrained Transformers, e.g., BERT [5], along with its many successors such as XLNet [30] and RoBERTa [13], have led to state-of-the-art performance on many tasks, such as question answering, part-of-speech tagging, information retrieval, and sentence classification with very few labels. Deep pretrained Transformer models induce powerful token-level and sentence-level embeddings that can be rapidly fine-tuned on many downstream NLP problems by adding a task-specific lightweight linear layer on top of the transformer models.

However, how to successfully apply Transformer models to XMC problems remains an open challenge, primarily due to the extremely large output space and severe label sparsity issues. As a concrete example, Table 1 compares the model size (in terms of the number of model parameters) and GPU memory usage, when applying a 24-layer XLNet model to a binary classification problem (e.g., the MNLI dataset of GLUE [27]) versus its application to an XMC problem with 1 million labels. Note that the classifier for the MNLI problem and XMC problem has a model size of 2K and 1025M, respectively. This means that the latter is a much harder problem than the former from the model optimization point of view. Additionally, in attempting to solve the XMC problem, we run out of GPU memory even for a single example mini-batch update. Table 1 gives the details of the GPU memory usage in the training stages of one forward pass, one backward pass and one optimization step, respectively.

In addition to the computational challenges, the large output space in XMC is exacerbated by a severe label sparsity issue. The left part of Figure 1 illustrates the “long-tailed” label distribution

	XLNet-large model (# params)			(batch size, sequence length)=(1,128)			
problem	encoder	classifier	total	load model	+forward	+backward	+optimizer step
GLUE (MNLI)	361 M	2 K	361 M	2169 MB	2609 MB	3809 MB	6571 MB
XMC (1M)	361 M	1,025 M	1,386 M	6077 MB	6537 MB	OOM	OOM

Table 1: On the left of are the model sizes (numbers of parameters) when applying the XLNet-large model to the MNLI problem vs. the XMC (1M) problem; on the right is the GPU memory usage (in megabytes) in solving the two problems, respectively. The results were obtained on a recent Nvidia 2080Ti GPU with 12GB memory. OOM stands for out-of-memory.

in the Wiki-500K data set [25]. Only 2% of the labels have more than 100 training instances, while the remaining 98% are long-tail labels with much fewer training instances. How to successfully fine-tune Transformer models with such sparsely labeled data is a tough question that has not been well-studied so far, to the best of our knowledge.

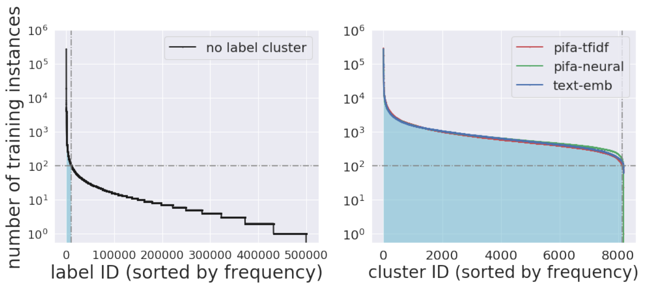


Figure 1: On the left, Wiki-500K **shows a long-tail distribution of labels. Only** 2.1% **of the labels have more than 100 training instances, as indicated by the cyan blue regime. On the right is the clusters distribution after our semantic label indexing based on different label representations;** 99.4% **of the clusters have more than 100 training instances, which mitigates the data sparsity issue for fine-tuning of Transformer models.**

Instead of fine-tuning deep Transformer models and dealing with the bottleneck classifier layer, an alternative is to use a more economical transfer learning paradigm as studied in the context of word2vec [15], ELMo [19], and GPT [22]. For instance, ELMo uses a (bi-directional LSTM) model pretrained on large unlabeled text data to obtain contexualized word embeddings. When applying ELMo on a downstream task, these word embeddings can be used as input without adaptation. This is equivalent to freezing the ELMo encoder, and fine-tuning the downstream task-specific model on top of ELMo, which is much more efficient in terms of memory as well as computation. However, such a benefit comes at the price of limiting the model capacity from adapting the encoder, as we will see in the experimental results in Section 4.

In this paper, we propose X-Transformer, a new approach that overcomes the aforementioned issues, with successful fine-tuning of deep Transformer models for the XMC problem. X-Transformer consists of a Semantic Label Indexing component, a Deep Neural Matching component, and an Ensemble Ranking component. First, Semantic label Indexing (SLI) decomposes the original intractable XMC problem into a set of feasible sub-problems of much smaller output space via label clustering, which mitigates the label sparsity issue as shown in the right part of Figure 1. Second, the Deep Neural Matching component fine-tunes a Transformer model for each of the SLI-induced XMC sub-problems, resulting in a better mapping from the input text to the set of label clusters. Finally, the Ensemble Ranking component is trained conditionally on the instance-to-cluster assignment and neural embedding from the Transformer, and is used to assemble scores derived from various SLI-induced sub-problems for further performance improvement.

In our experiments, the proposed X-Transformer achieves new state-of-the-art results on four XMC benchmarks and leads to improvement on two real-would XMC applications. On a Wiki dataset with a half million labels, the precision@1 of X-Transformer reaches 77.28%, a substantial improvement over the well-established hierarchical label tree approach Parabel [20] (i.e., 68.70%) and the competing deep learning method AttentionXML [32] (i.e., 76.95%). Furthermore, X-Transformer also demonstrates great impact on the scalability of deep Transformer models in real-world large applications. In our application of X-Transformer to Amazon Product2Query problem that can be formulated as XMC, X-Transformer significantly outperforms Parabel too. The dataset, experiment code, models are available: https://github.com/OctoberChang/X-Transformer.

2 RELATED WORK AND BACKGROUND

2.1 Extreme Multi-label Classification

Sparse Linear Models. To overcome computational issues, most existing XMC algorithms use sparse TF-IDF features (or slight variants), and leverage different partitioning techniques on the label space to reduce complexity. For example, sparse linear one-vs-all (OVA) methods such as DiSMEC [1], ProXML [2] and PPDSparse [31] explore parallelism to speed up the algorithm and reduce the model size by truncating model weights to encourage sparsity. OVA approaches are also widely used as building blocks for many other approaches, for example, in Parabel [20] and SLICE [7], linear OVA classifiers with smaller output domains are used.

The efficiency and scalability of sparse linear models can be further improved by incorporating different partitioning techniques on the label spaces. For instance, Parabel [20] partitions the labels through a balanced 2-means label tree using label features constructed from the instances. Recently, several approaches are proposed to improve Parabel. Bonsai [9] relaxes two main constraints in Parabel: 1) allowing multi-way instead of binary partitionings of the label set at each intermediate node, and 2) removing strict balancing constraints on the partitions. SLICE [7] considers building an approximate nearest neighbor (ANN) graph as an indexing structure over the labels. For a given instance, the relevant labels can be found quickly from the nearest neighbors of the instance via the ANN graph.

Deep Learning Approaches. Instead of using handcrafted TF-IDF features which are hard to optimize for different downstream XMC problems, deep learning approaches employ various neural network architectures to extract semantic embeddings of the input text. XML-CNN [12] employs one-dimensional Convolutional neural networks along both sequence length and word embedding dimension for representing text input. As a follow-up, SLICE considers dense embedding from the supervised pre-trained XML-CNN models as the input to its hierarchical linear models. More recently, AttentionXML [32] uses BiLSTMs and label-aware attention as the scoring function, and performs warm-up training of the models with hierarchical label trees. In addition, AttentionXML consider various negative sampling strategies on the label space to avoid back-propagating the entire bottleneck classifier layer.

2.2 Transfer Learning Approaches in NLP

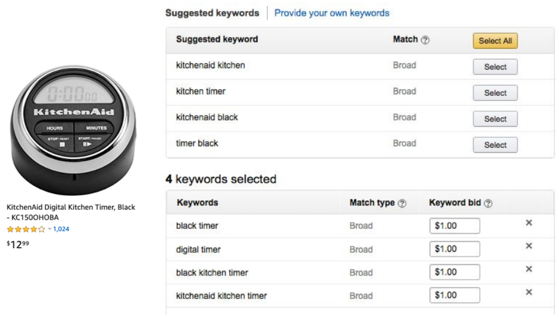
Recently, the NLP community has witnessed a dramatic paradigm shift towards the "pre-training then fine-tuning" framework. One of the pioneering works is BERT [5], whose pre-training objectives are masked token prediction and next sentence prediction tasks. After pre-training on large-scale unsupervised corpora such as Wikipedia and BookCorpus, the Transformer model demonstrates vast improvement over existing state-of-the-art when fine-tuned on many NLP tasks such as the GLUE benchmark [27], named entity recognition, and question answering. More advanced variants of the pre-trained Transformer models include XLNet [30] and RoBERTa [13]. XLNet considers permutation language modeling as the pre-training objective and two-stream self-attention for target-aware token prediction. It is worth noting that the contextualized token embeddings extracted from XLNet also demonstrate competitive performance when fed into a task-specific downstream model on large-scale retrieval problems. RoBERTa improves upon BERT by using more robust optimization with large-batch size update, and pre-training the model for longer till it truly converges.

However, transferring the success of these pre-trained Transformer models on the GLUE text classification to the XMC problem is non-trivial, as we illustrated in Table 1. Before the emergence of BERT-type end-to-end fine-tuning, the canonical way of transfer learning in NLP perhaps comes from the well-known Word2Vec [15] or GloVe [18] papers. Word2vec is a shallow two-layer neural network that is trained to reconstruct the linguistic context of words. GLoVe considers a matrix factorization objective to reconstruct the global word-to-word co-occurrence in the corpus. A critical downside of Word2vec and GloVe is that the pre-trained word embeddings are not contextualized depending on the local surrounding word. ELMo [19] and GPT2 [22] instead present contextualized word embeddings by using large BiLSTM or Transformer models. After the models are pre-trained, transfer learning can be easily carried out by feeding these extracted word embeddings as input to the downstream task-specific models. This is more efficient compared to the BERT-like end-to-end additional fine-tuning of the encoder, but comes at the expense of losing model expressiveness. In the experimental results section, we show that using fixed word embeddings from universal pre-trained models such as BERT is not powerful enough for XMC problems.

2.3 Amazon Applications

Many challenging problems at Amazon amount to finding relevant results from an enormous output space of potential candidates: for example, suggesting keywords to advertisers starting new campaigns on Amazon, predicting next queries a customer will type based on the previous queries he/she typed. Here we discuss keyword recommendation system for Amazon Sponsored Products, as illustrations in Fig.2, and how it can be formulated as XMC problems.

Keyword recommendation system. Keyword Recommendation Systems provide keyword suggestions for advertisers to create campaigns. In order to maximize the return of investment for the advertisers, the suggested keywords should be highly relevant to their products so that the suggestions can lead to conversion. An XMC model, when trained on an product-to-query dataset such as product-query customer purchase records, can suggest queries that are relevant to any given product by utilizing product information, like title, description, brand, etc.



KitchenAid Digital Kitchen Timer, Black - KC150OHOBA
1,024
$12.99

Suggested keyword	Match
kitchenaid kitchen	Broad
kitchen timer	Broad
kitchenaid black	Broad
timer black	Broad

4 keywords selected

Keywords	Match type	Keyword bid
black timer	Broad	$1.00
digital timer	Broad	$1.00
black kitchen timer	Broad	$1.00
kitchenaid kitchen timer	Broad	$1.00

Figure 2: keyword recommendation system

3 PROPOSED METHOD: X-TRANSFORMER

3.1 Problem Formulation

Motivations. Given a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \{0,1\}^L, i = 1, \ldots, N\}$, extreme multi-label classification aims to learn a scoring function f that maps an input (or instance) $\mathbf{x}_i$ and a label l to a score $f(\mathbf{x}_i, l) \in \mathbb{R}$. The function f should be optimized such that the score is high when $y_{il} = 1$ (i.e., label l is relevant to instance $\mathbf{x}_i$) and the score is low when $y_{il} = 0$. A simple one-versus-all approach realizes the scoring function f as

$$f(\mathbf{x}, l) = \mathbf{w}_l^T \phi(\mathbf{x})$$

where $\phi(\mathbf{x})$ represents an encoding and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_L]^T \in \mathbb{R}^{L \times d}$ is the classifier bottleneck layer. For convenience, we further define the top-b prediction operator as

$$f_b(\mathbf{x}) = \text{Top-b}\Big(\big[f(\mathbf{x}, 1), \ldots, f(\mathbf{x}, L)\big]\Big) \in \{1, \ldots, L\},$$

where $f_b(\mathbf{x})$ is an index set containing the top-b predicted labels. As we pointed out in Table 1, it is not only very difficult to fine-tune the Transformer encoders $\phi_T(\mathbf{x}; \theta)$ together with the intractable classifier layer $\mathbf{W}$, but also extremely slow to compute the top-K predicted labels efficiently.

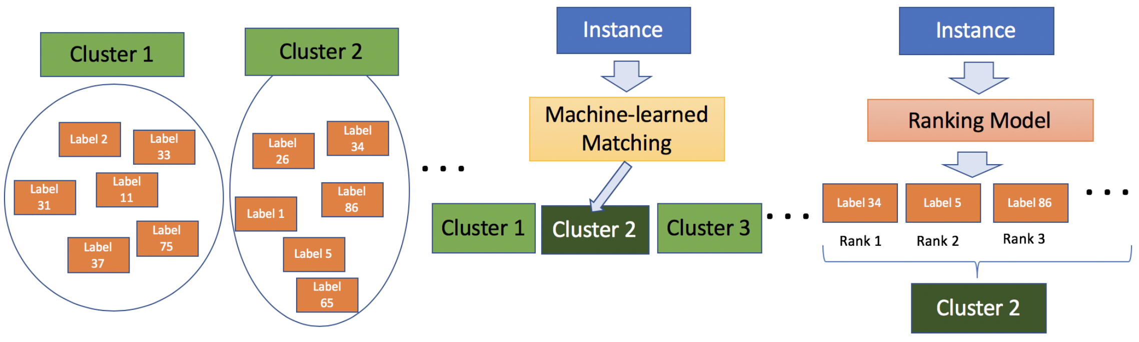


Figure 3: The proposed X-Transformer framework. First, Semantic Label Indexing reduces the large output space. Transformers are then fine-tuned on the XMC sub-problem that maps instances to label clusters. Finally, linear rankers are trained conditionally on the clusters and Transformer's output in order to re-rank the labels within the predicted clusters.

High-level Sketch. To this end, we propose X-Transformer as a practical solution to fine-tune deep Transformer models on XMC problems. Figure 3 summarizes our proposed framework.

In a nutshell, X-Transformer decomposes the intractable XMC problem to a feasible sub-problem with a smaller output space, which is induced from semantic label indexing, which clusters the labels. We refer to this sub-problem as the neural matcher of the following form:

$$g(\mathbf{x}, k) = \mathbf{w}_k^T \phi_T(\mathbf{x}), \quad k = 1, \ldots, K \tag{1}$$

where K is the number of clusters which is significantly smaller than the original intractable XMC problem of size $O(L)$. Finally, X-Transformer currently uses a linear ranker that conditionally depends on the embedding of transformer models and its top-b predicted clusters $g_b(\mathbf{x})$.

$$f(\mathbf{x}, l) = \begin{cases} \sigma\big(g(\mathbf{x}, c_l), h(\mathbf{x}, l)\big), & \text{if } c_l \in g_b(\mathbf{x}), \\ -\infty, & \text{otherwise.} \end{cases} \tag{2}$$

Here $c_l \in \{1, \ldots, K\}$ represents the cluster index of label l, $g(\mathbf{x}, c_l)$ is the neural matcher realized by deep pre-trained Transformers, $h(\mathbf{x}, l)$ is the linear ranker, and $\sigma()$ is a non-linear activation function to combine the final scores from g and h. We now further introduce each of these three components in detail.

3.2 Semantic Label Indexing

Inducing latent clusters with semantic meaning brings several advantages to our framework. We can perform a clustering of labels that can be represented by a label-to-cluster assignment matrix $\mathrm{C} \in \{0, 1\}^{L \times K}$ where $c_{lk} = 1$ means label l belongs to cluster k. The number of clusters K is typically set to be much smaller than the original label space L. Deep Transformer models are fine-tuned on the induced XMC sub-problem where the output space is of size K, which significantly reduces the computational cost and avoids the label sparsity issue in Figure 1. Furthermore, the label clustering also plays a crucial role in the linear ranker $h(\mathbf{x}, l)$. For example, only labels within a cluster are used to construct negative instances for training the ranker. In prediction, ranking is only performed for labels within a few clusters predicted by our deep Transformer models.

Given a label representation, we cluster the L labels hierarchically to form a hierarchical label tree with K leaf nodes [7, 9, 20, 32]. For simplicity, we consider binary balanced hierarchical trees [14, 20] as the default setting. Due to the lack of a direct and informative representation of the labels, the indexing system for XMC may be noisy. Fortunately, the instances in XMC are typically very informative. Therefore, we can utilize the rich information of the instances to build a strong matching system as well as a strong ranker to compensate for the indexing system.

Label embedding via label text. Given text information about labels, such as a short description of categories in the Wikipedia dataset or search queries on the Amazon shopping website, we can use this short text to represent the labels. In this work, we use a pretrained XLNet [19] to represent the words in the label. The label embedding is the mean pooling of all XLNet word embeddings in the label text. Specifically, the label embedding of label l is

$$\psi_{\text{text-emb}}(l) = \frac{1}{|\text{text}(l)|} \sum_{w \in text(l)} \phi_{\text{xlnet}}(w)$$

where $\phi_{xlnet}(w)$ is the hidden embedding of token w in label l.

Label embedding via embedding of positive instances. The short text of labels may not contain sufficient information and is often ambiguous and noisy for some XMC datasets. Therefore we can derive a label representation from embedding of its positive instances. Specifically, the label embedding of label l is

$$\psi_{\text{pifa-tfidf}}(l) = \mathbf{v}_l / \|\mathbf{v}_l\|, \ \mathbf{v}_l = \sum_{i:y_{il}=1} \phi_{\text{tf-idf}}(\mathbf{x}_i), \quad l = 1, \ldots, L,$$
$$\psi_{\text{pifa-neural}}(l) = \mathbf{v}_l / \|\mathbf{v}_l\|, \ \mathbf{v}_l = \sum_{i:y_{il}=1} \phi_{\text{xlnet}}(\mathbf{x}_i), \quad l = 1, \ldots, L.$$

We refer to this type of label embedding as Positive Instance Feature Aggregation (PIFA), which is used in recent state-of-the-art XMC methods [7, 9, 20, 32]. Note that X-Transformer is not limited by the above mentioned label representations; indeed in applications where labels encode richer meta information such as a graph, we can use label representations derived from graph clustering and graph convolution.

3.3 Deep Transformer as Neural Matcher

After Semantic Label Indexing (SLI), the original intractable XMC problem morphs to a feasible XMC sub-problem with a much smaller output space of size K. See Table 2 for the exact K that we used for each XMC data set. Specifically, the deep Transformer model now aims to map each text instance to the assigned relevant clusters. The induced instance-to-cluster assignment matrix is $\mathbf{M} = \mathbf{YC} = [\mathbf{m}_1, \ldots, \mathbf{m}_i, \ldots, \mathbf{m}_N]^T \in \{0,1\}^{N \times K}$ where $\mathbf{Y} \in \mathbb{R}^{N \times L}$ is the original instance-to-label assignment matrix and $\mathbf{C} \in \mathbb{R}^{L \times K}$ is the label-to-cluster assignment matrix provided by the SLI stage. The goal now becomes fine-tuning deep Transformer models $g(\mathbf{x}, k; \mathbf{W}, \theta)$ on $\{(\mathbf{x}_i, \mathbf{m}_i) | i = 1, \ldots, N\}$ such that

$$\min_{\mathbf{W}, \theta} \quad \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \max\left(0, 1 - \tilde{M}_{ik} g(\mathbf{x}, k; \mathbf{W}, \theta)\right)^2, \tag{3}$$
$$\text{s.t.} \quad g(\mathbf{x}, k; \mathbf{W}, \theta) = \mathbf{w}_k^T \phi_{\text{transformer}}(\mathbf{x}),$$

where $\tilde{\mathbf{M}}_{ik} = 2\mathbf{M}_{ik} - 1 \in \{-1, 1\}$, $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K]^T \in \mathbb{R}^{K \times d}$, and $\phi_{\text{transformer}}(\mathbf{x}) \in \mathbb{R}^d$ is the embedding from the Transformers. We use the squared-hinge loss in the matching objective (3) as it has shown better ranking performance as shown in [31]. Next, we discuss engineering optimizations and implementation details that considerably improve training efficiency and model performance.

Pretrained Transformers. We consider three state-of-the-art pre-trained Transformer-large-cased models (i.e., 24 layers with case-sensitive vocabulary) to fine-tune, namely BERT [5], XLNet [30], and RoBERTa [13]. The instance embedding $\phi(\mathbf{x})$ is the "[CLS]"-like hidden states from the last layer of BERT, RoBERTa and XLNet. Computationally speaking, BERT and RoBERTa are similar while XLNet is nearly 1.8 times slower. In terms of performance on XMC tasks, we found RoBERTa and XLNet to be slightly better than BERT, but the gap is not as significant as in the GLUE benchmark. More concrete analysis is available in Section 4.

It is possible to use Automatic Mixed Precision (AMP) between Float32 and Float16 for model fine-tuning, which can considerably reduce the model's GPU memory usage and training speed. However, we used Float32 for all the experiments as our initial trials of training Transformers in AMP mode often led to unstable numerical results for the large-scale XMC dataset Wiki-500K.

Input Sequence Length. The time and space complexity of the Transformer scales quadratically with the input sequence length, i.e., $O(T^2)$ [26], where $T = \text{len}(\mathbf{x})$ is the number of tokenized sub-words in the instance $\mathbf{x}$. Using smaller T reduces not only the GPU memory usage that supports using larger batch size, but also increases the training speed. For example, BERT first pre-trains on inputs of sequence length 128 for 90% of the optimization, and the remaining 10% of optimization steps on inputs of sequence length 512 [5]. Interestingly, we observe that the model fine-tuned with sequence length 128 v.s. sequence length 512 does not differ significantly in the downstream XMC ranking performance. Thus, we fix the input sequence length to be $T = 128$ for model fine-tuning, which significantly speeds up the training time. It would be interesting to see if we can bootstrap training the Transformer models from shorter sequence length and ramp up to larger sequence length (e.g., 32, 64, 128, 256), but we leave that as future work.

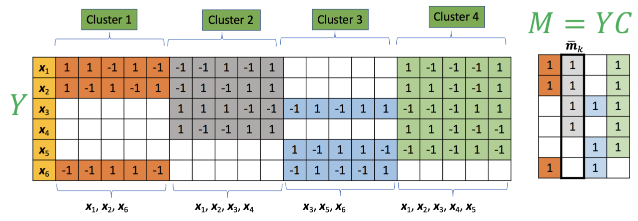


Figure 4: Training rankers with the Teacher Forcing Negatives(TFN) strategy. For illustration, we have $N = 6$ instances, $L = 20$ labels, $K = 4$ label clusters, and $\mathrm{M} \in \{0,1\}^{6 \times 4}$ denotes the instance-to-cluster assignment matrix. For example, Cluster 1 with the orange color contains the first 5 labels. The nonzeros of the first column of M correspond to $\{x_1, x_2, x_6\}$, which are instances with at least one positive label contained in Cluster 1. For each label in the first cluster, the ranker using Teacher Forcing Negatives (TFN) only considers these three instances. Matcher-aware Negatives (MAN) strategy is introduced in Section 3.4 to further add improved hard negatives to enhance the TFN strategy.

Bootstrapping Label Clustering and Ranking. After fine-tuning a deep Transformer model, we have powerful instance representation $\phi_{\text{transformer}}(\mathbf{x})$ that can be used to bootstrap semantic label clustering and ranking. For label clustering, the embedding label l can be constructed by aggregating the embeddings of its positive instances. For ranking, the fine-tuned Transformer embedding can be concatenated with the sparse TF-IDF vector for better modeling power. See details in the ablation study Table 5.

3.4 Ranking

After the matching step, a small subset of label clusters is retrieved. The goal of the ranker is to model the relevance between the instance and the labels from the retrieved clusters. Formally, given a label l and an instance $\mathbf{x}$, we use a linear one-vs-all (OVA) classifier to parameterize the ranker $h(\mathbf{x}, l) = \mathbf{w}_l^T \phi(\mathbf{x})$ and train it with a binary loss. For each label, naively estimating the weights $\mathbf{w}_l$ based on all instances $\{(\mathbf{x}_i, Y_{i,l})\}_{i=1}^N$ takes $O(N)$, which is too expensive. Instead, we consider two sampling strategies that only include hard negative instances to reduce the computational complexity: Teacher Forcing Negatives (TFN) and Matcher-aware Negatives (MAN).

Teacher Forcing Negatives (TFN). for each label l, we only include a subset of instances induced by the instance-to-cluster assignment matrix $\mathbf{M} = \mathbf{YC}$. In particular, in addition to the positive instances corresponding to the l-th label, we only include instances whose labels belong to the same cluster as the l-th label, i.e., $\{(\mathbf{x}_i, y_{i,l} : i \in \{i : \mathbf{M}_{i,c_l} = 1\}\}$. In Figure 4, we illustrate the TFN strategy with a toy example. As the first five labels belong to Cluster 1, and only $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_6\}$ contain a positive label within this cluster, we only consider this subset of instances to train a binary classifier for each of the first five labels.

Matcher-aware Negatives (MAN). The Teacher Forcing strategy only includes negative instances which are *hard* from the "teacher", i.e., the ground truth instance-to-clustering assignment matrix $\mathbf{M}$ used to train our neural matcher. However, $\mathbf{M}$ is independent from the performance of our neural matcher. Thus, training ranker with the TFN strategy alone might introduce an exposure bias issue, i.e., training-inference discrepancy. Instead, we also consider including matcher-aware hard negatives for each label. In particular, we can use the instance-to-cluster prediction matrix $\hat{\mathbf{M}} \in \{0,1\}^{N \times K}$ from our neural matcher, where the nonzeros of the i-th row of $\hat{\mathbf{M}}$ correspond to the top-b predicted clusters from $g_b(\mathbf{x}_i)$. In practice, we observe that a combination of TFN and MAN yields the best performance, i.e., using $\mathbf{M}' = \mathbf{YC} + \hat{\mathbf{M}}$ to include hard negatives for each label. See Table 5 for a detailed Ablation study.

For the ranker input representation, we not only leverage the TF-IDF features $\phi_{\text{tf-idf}}(\mathbf{x})$, but also exploit the neural embeddings $\phi_{\text{neural}}(\mathbf{x})$ from either the pre-trained or fine-tuned Transformer model. After the ranker is trained, the final ranking scores are computed via (2). We can further ensemble the scores from different X-Transformer models, which are trained on different semantic-aware label clusters or different pre-trained Transformer models such as BERT, RoBERTa and XLNet.

4 EMPIRICAL RESULTS

The experiment code, including datasets and fine-tuned models are publicly available. [1]

4.1 Datasets and Preprocessing

XMC Benchmark Data. We consider four multi-label text classification data sets used in AttentionXML [32] for which we had access to the raw text representation, namely Eurlex-4K, Wiki10-31K, AmazonCat-13K and Wiki-500K. Summary statistics of the data sets are given in Table 2. We follow the training and test split of [32] and set aside 10% of the training instances as the validation set for hyperparameter tuning.

Amazon Applications. We consider an internal Amazon data set, namely Prod2Query-1M, which consists of 14 million instances (products) and 1 million labels (queries) where the label is positive if a product is clicked at least once as a result of a search query. We divide the data set into 12.5 million training samples, 0.8 million validation samples and 0.7 million testing samples.

4.2 Algorithms and Hyperparameters

Comparing Methods. We compare our proposed X-Transformer method to the most representative and state-of-the-art XMC methods including the embedding-based AnnexML [24]; one-versus-all DiSMEC [1]; instance tree based PfastreXML [8]; label tree based Parabel [20], eXtremeText [29], Bonsai [9]; and deep learning based XML-CNN [12], AttentionXML [32] methods. The results of all these baseline methods are obtained from [32, Table 3]. For evaluation with other XMC approaches that have not released their code or are difficult to reproduce, we have a detailed comparison in Table 6.

Evaluation Metrics. We evaluate all methods with example-based ranking measures including Precision@k ($k = 1, 3, 5$) and Recall@k ($k = 1, 3, 5$), which are widely used in the XMC literature [3, 8, 20, 21, 23].

Hyperparameters. For X-Transformer, all hyperparameters are chosen from the held-out validation set. The number of clusters are listed in Table 2, which are consistent with the Parabel setting for fair comparison. We consider the 24 layers cased models of BERT [5], RoBERTa [13], and XLNet [30] using the Pytorch implementation from HuggingFace Transformers [28][2]. For fine-tuning the Transformer models, we set the input sequence length to be 128 for efficiency, and the batch size per GPU to be 16 along with gradient accumulation step of 4, and use 4 GPUs per model. This together amounts to a batch size of 256 in total. We use Adam [10] with linear warmup scheduling as the optimizer where the learning rate is chosen from $\{4, 5, 6, 8\} \times 10^{-5}$. Models are trained until convergence, which takes 1k, 1.4k, 20k, 50k optimization steps for Eurlex-4K, Wiki10-31K, AmazonCat-13K, Wiki-500K, respectively.

4.3 Results on Public XMC Benchmark Data

Table 3 compares the proposed X-Transformer with the most representative SOTA XMC methods on four benchmark datasets. Following previous XMC works, we focus on top predictions by presenting Precision@k, where $k = 1, 3, 5$.

The proposed X-Transformer outperforms all XMC methods, except being slightly worse than AttentionXML in terms of P@3 and P@5 on the Wiki-500K dataset. We also compare X-Transformer against linear baselines using Parabel model with three different input representations: (1) $\phi_{\text{pre-xlnet}}$ denotes pretrained XLNet embeddings (2) ϕ_{tfidf} denotes TF-IDF embeddings (3) $\phi_{\text{fnt-xlnet}} \oplus \phi_{\text{tfidf}}$ denotes finetuned XLNet embeddings concatenated with TF-IDF embeedings. We clearly see that the performance of baseline (1) is significantly worse. This suggests that the ELMo-style transfer learning, though efficient, is not powerful to achieve good performance for XMC problems. The performance of baseline (2) is similar to that of Parabel, while baseline (3) further improves performance due to the use of fine-tuned XLNet embeddings.

AttentionXML [32] is a very recent deep learning method that uses BiLSTM and label-aware attention layer to model the scoring function. They also leverage hierarchical label trees to recursively warm-start the models and use hard negative sampling techniques to avoid using the entire classifier bottleneck layer. Some of the techniques in AttentionXML are complementary to our proposed X-Transformer, and it would be interesting to see how X-Transformer can be improved from those techniques.

4.4 Results on Amazon Applications.

Recall that the Amazon data consists of 12 million products and 1 million queries along with product-query relevance. We treat queries as output labels and product title as input. We use the default Parabel method (using TFIDF features) as the baseline method and show X-Transformer's relative improvement of precision and recall over the baseline in Table 4.

4.5 Ablation Study

We carefully conduct an ablation study of X-Transformer as shown in Table 5. We analyze the X-Transformer framework in terms of its four components: indexing, matching, ranker input representation, and training negative-sampling training algorithm. The configuration Index 9 represents the final best configuration as reported

[1]https://github.com/OctoberChang/X-Transformer

[2]https://github.com/huggingface/transformers

Dataset	n_{trn}	n_{tst}	$\|D_{\text{trn}}\|$	$\|D_{\text{trn}}\|$	L	$\bar{L}$	$\bar{n}$	K
Eurlex-4K	15,449	3,865	19,166,707	4,741,799	3,956	5.30	20.79	64
Wiki10-31K	14,146	6,616	29,603,208	13,513,133	30,938	18.64	8.52	512
AmazonCat-13K	1,186,239	306,782	250,940,894	64,755,034	13,330	5.04	448.57	256
Wiki-500K	1,779,881	769,421	1,463,197,965	632,463,513	501,070	4.75	16.86	8192

Table 2: Data Statistics. n_{trn}, n_{tst} refer to the number of instances in the training and test sets, respectively. $|D_{\text{trn}}|, |D_{\text{tst}}|$ refer to the number of word tokens in the training and test corpus, respectively. L is the number of labels, $\bar{L}$ the average number of labels per instance, $\bar{n}$ the average number of instances per label, and K is the number of clusters. The four benchmark datasets are the same as AttentionXML **[32] for fair comparison.**

Methods	Prec@1	Prec@3	Prec@5	Methods	Prec@1	Prec@3	Prec@5
Eurlex-4K				Wiki10-31K			
AnnexML [24]	79.66	64.94	53.52	AnnexML [24]	86.46	74.28	64.20
DiSMEC [1]	83.21	70.39	58.73	DiSMEC [1]	84.13	74.72	65.94
PfastreXML [8]	73.14	60.16	50.54	PfastreXML [8]	83.57	68.61	59.10
Parabel [20]	82.12	68.91	57.89	Parabel [20]	84.19	72.46	63.37
eXtremeText [29]	79.17	66.80	56.09	eXtremeText [29]	83.66	73.28	64.51
Bonsai [9]	82.30	69.55	58.35	Bonsai [9]	84.52	73.76	64.69
MLC2seq [16]	62.77	59.06	51.32	MLC2seq [16]	80.79	58.59	54.66
XML-CNN [12]	75.32	60.14	49.21	XML-CNN [12]	81.41	66.23	56.11
AttentionXML [32]	87.12	73.99	61.92	AttentionXML [32]	87.47	78.48	69.37
$\phi_{\text{pre-xlnet}}$ + Parabel	33.53	26.71	22.15	$\phi_{\text{pre-xlnet}}$ + Parabel	81.77	64.86	54.49
ϕ_{tfidf} + Parabel	81.71	69.15	58.11	ϕ_{tfidf} + Parabel	84.27	73.20	63.66
$\phi_{\text{fnt-xlnet}} \oplus \phi_{\text{tfidf}}$ + Parabel	84.09	71.50	60.12	$\phi_{\text{fnt-xlnet}} \oplus \phi_{\text{tfidf}}$ + Parabel	87.35	78.24	68.62
X-Transformer	**87.22**	**75.12**	**62.90**	X-Transformer	**88.51**	**78.71**	**69.62**
AmazonCat-13K				Wiki-500K			
AnnexML [24]	93.54	78.36	63.30	AnnexML [24]	64.22	43.15	32.79
DiSMEC [1]	93.81	79.08	64.06	DiSMEC [1]	70.21	50.57	39.68
PfastreXML [8]	91.75	77.97	63.68	PfastreXML [8]	56.25	37.32	28.16
Parabel [20]	93.02	79.14	64.51	Parabel [20]	68.70	49.57	38.64
eXtremeText [29]	92.50	78.12	63.51	eXtremeText [29]	65.17	46.32	36.15
Bonsai [9]	92.98	79.13	64.46	Bonsai [9]	69.26	49.80	38.83
MLC2seq [16]	94.26	69.45	57.55	MLC2seq [16]	-	-	-
XML-CNN [12]	93.26	77.06	61.40	XML-CNN [12]	-	-	-
AttentionXML [32]	95.92	82.41	67.31	AttentionXML [32]	76.95	**58.42**	**46.14**
$\phi_{\text{pre-xlnet}}$ + Parabel	80.96	63.92	50.72	$\phi_{\text{pre-xlnet}}$ + Parabel	31.83	20.24	15.76
ϕ_{tfidf} + Parabel	92.81	78.99	64.31	ϕ_{tfidf} + Parabel	68.75	49.54	38.92
$\phi_{\text{fnt-xlnet}} \oplus \phi_{\text{tfidf}}$ + Parabel	95.33	82.77	67.66	$\phi_{\text{fnt-xlnet}} \oplus \phi_{\text{tfidf}}$ + Parabel	75.57	55.12	43.31
X-Transformer	**96.70**	**83.85**	**68.58**	X-Transformer	**77.28**	57.47	45.31

Table 3: Comparing X-Transformer **against state-of-the-art XMC methods on** Eurlex-4K, Wiki10-31K, AmazonCat-13K, **and** Wiki-500K. **The baselines' results are from [32, Table 3]. Note that** MLC2seq **and** XML-CNN **are not scalable on** Wiki-500K. **We also present linear baselines (**Parabel**) with three input representations. Specifically, $\phi_{\text{pre-xlnet}}$ denotes pre-trained XLNet embeddings, ϕ_{tfidf} denotes TF-IDF embeddings, $\phi_{\text{fnt-xlnet}} \oplus \phi_{\text{tfidf}}$ denotes fine-tuned XLNet embeddings concatenate with TF-IDF embeddings.**

	Precision			Recall		
Methods	@1	@5	@10	@1	@5	@10
X-Transformer	**10.7%**	**7.4%**	**6.6%**	**12.0%**	**4.9%**	**2.8%**

Table 4: Relative improvement over Parabel **on the Prod2Query data set.**

in Table 3. There are four takeaway messages from this ablation study, and we describe them in the following four paragraphs.

Ranker Representation and Training. Config. ID 0, 1, 2 shows the effect of input representation and training strategy for the ranking. The benefit of using instance embedding from fine-tuned transformers can be seen from config. ID 0 to 1. In addition, from ID 1 to 2, we observe that using Teacher Forcing Negatives (TFN) is not enough for training the ranker, as it could suffer from the exposure

Dataset	Config. ID	X-Transformer Ablation Configuration				Evaluation Metric					
		indexing	matching	ranker input	negative-sampling	P@1	P@3	P@5	R@1	R@3	R@5
Eurlex-4K	0	pifa-tfidf	BERT	$\phi_{\text{tfidf}}(\mathbf{x})$	TFN	83.93	70.59	58.69	17.05	42.08	57.14
	1	pifa-tfidf	BERT	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN	85.02	71.83	59.87	17.21	42.79	58.30
	2	pifa-tfidf	BERT	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	85.51	72.95	60.83	17.32	43.45	59.21
	3	pifa-tfidf	RoBERTa	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	85.33	72.89	60.79	17.32	43.39	59.16
	4	pifa-tfidf	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	85.07	72.75	60.69	17.25	43.29	59.01
	5	pifa-neural	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	84.81	72.39	60.38	17.19	42.98	58.70
	6	text-emb	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	85.25	72.76	60.20	17.29	43.25	58.54
	7	all	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	86.55	74.24	61.96	17.54	44.16	60.24
	8	pifa-neural	all	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	85.92	73.43	61.53	17.40	43.69	59.86
	9	all	all	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	**87.22**	**75.12**	**62.90**	**17.69**	**44.73**	**61.17**
Wiki-500K	0	pifa-tfidf	BERT	$\phi_{\text{tfidf}}(\mathbf{x})$	TFN	69.52	49.87	38.71	22.30	40.62	48.65
	1	pifa-tfidf	BERT	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN	71.90	51.58	40.10	23.27	42.14	50.42
	2	pifa-tfidf	BERT	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	74.68	53.64	41.50	24.56	44.26	52.50
	3	pifa-tfidf	RoBERTa	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	75.40	54.32	42.06	24.85	44.93	53.30
	4	pifa-tfidf	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	75.45	54.50	42.24	24.81	45.00	53.44
	5	pifa-neural	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	76.34	55.50	43.04	25.15	45.88	54.53
	6	text-emb	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	74.12	52.85	40.53	24.18	43.30	50.98
	7	all	XLNet	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	75.85	56.08	44.24	24.80	46.36	56.35
	8	pifa-neural	all	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	**77.44**	56.84	44.37	25.61	47.18	56.55
	9	all	all	$\phi_{\text{tfidf}}(\mathbf{x}) \oplus \phi_{\text{neural}}(\mathbf{x})$	TFN + MAN	77.28	**57.47**	**45.31**	**25.48**	**47.82**	**57.95**

Table 5: Ablation study of X-Transformer **on** Eurlex-4K **and** Wiki-500K **data sets. We outline four take away messages: (1) Config. ID=** $\{0, 1, 2\}$ **demonstrates better performance by using Matcher-aware Negatives (MAN) and Neural embedding for training the rankers; (2) Config. ID=** $\{2, 3, 4\}$ **suggests that, performance-wise, XLNet is similar to RoBERTa, and slightly better than BERT; (3) Config. ID=**$\{4, 5, 6\}$ **manifests the importance of label clusters induced from different label representations. (4) Config. ID=**$\{7, 8, 9\}$ **indicates the effect of ensembling various configuration of the models.**

bias of only using the ground truth clustering assignment, but ignores the hard negatives mistakenly produced by the Transformer models. Note that techniques such as adding Matcher-aware negatives (MAN) from previous model's prediction to bootstrap the next level's model training is also used in AttentionXML [32].

Different Transformer Models. Next, we analyze how the three different Transformer models (i.e., BERT, RoBERTa, XLNet) affect the performance, as shown in Config. ID 2, 3, 4. For Wiki-500K, we observe that the XLNet and RoBERTa are generally more powerful than the BERT models. On the other hand, such an advantage is not clear for Eurlex-4K, possibly due to the nature of the data set.

Label Representation for Clustering. The importance of different label representation for clustering is demonstrated in Config. ID 4, 5, 6. For Eurlex-4K, we see that using label text embedding as representation (i.e. text-emb) leads to the strong performance compared to pifa-tfidf (id 4) and pifa-neural (id 5). In contrast, pifa-tfidf becomes the best performing representation on the Wiki-500K dataset. This phenomenon could be due to the label text of Wiki-500K being more noisy compared to Eurlex-4K, which deteriorates the label clustering results on Wiki-500K.

Ensemble Ranking. Finally, we show the advantage of ensembing prediction from different models as shown in Config. ID 7, 8, 9. For Eurlex-4K, combining predictions from different label representations (ID 7) is better than from different Transformer models (ID 8). Combining all (ID 9) leads to our final model, X-Transformer.

4.6 Cross-Paper Comparisons

Many XMC approaches have been proposed recently. However, it is sometimes difficult to compare metrics directly from different papers. For example, the P@1 of Parabel on Wiki-500K is 59.34% in [7, Table 2] and 68.52% in [20, Table 2], but we see 68.70% in Table 3. The inconsistency may be due to differences in data processing, input representation, or other reasons. We propose an approach to calibrate these numbers so that various methods can be compared in a more principled way. In particular, for each metric $m(\cdot)$, we use the relative improvement over a common anchor method, which is set to be Parabel as it is widely used in the literature. For a competing method X with a metric $m(\text{X})$ on a data set reported in a paper, we can compute the relative improvement over Parabel as follows: $\frac{m(\text{X})-m(\text{Parabel})}{m(\text{Parabel})} \times 100\%$, where $m(\text{Parabel})$ is the metric obtained by Parabel on the same data set in the same paper. Following the above approach, we include a variety of XMC approaches in our comparison. We report the relative improvement of various methods on two commonly used data sets, Eurlex-4K and Wiki-500K, in Table 6. We can clearly observe that X-Transformer brings the most significant improvement over Parabel and SLICE.

5 CONCLUSIONS

In this paper, we propose X-Transformer, the *first* scalable framework to fine-tune Deep Transformer models that improves state-of-the-art XMC methods on four XMC benchmark data sets. We further applied X-Transformer to a real-life application, product2query prediction, showing significant improvement over the competitive linear models, Parabel.

Eurlex-4K					Wiki-500K				
Method	Source	Relative Improvement over Parabel (%)			Method	Source	Relative Improvement over Parabel (%)		
		Prec@1	Prec@3	Prec@5			Prec@1	Prec@3	Prec@5
X-Transformer	Table 3	**+6.27%**	**+9.08%**	**+8.55%**	X-Transformer	Table 3	**+12.49%**	**+15.94%**	**+17.26%**
SLICE	[7, Table 2]	+4.27%	+3.34%	+3.11%	SLICE	[7, Table 2]	+5.53%	+7.02%	+7.56%
GLaS	[6, Table 3]	-5.18%	-5.48%	-5.34%	GLaS	[6, Table 3]	+4.77%	+3.37%	+4.27%
ProXML	[2, Table 5]	+3.86%	+2.90%	+2.43%	ProXML	[2, Table 5]	+2.22%	+0.82%	+ 2.92%
PPD-Sparse	[20, Table 2]	+1.92%	+2.93%	+2.92%	PPD-Sparse	[20, Table 2]	+2.39%	+2.33%	+ 2.88%
SLEEC	[9, Table 2]	-3.53%	-6.40%	-9.04%	SLEEC	[9, Table 2]	-29.84%	-40.73%	-45.08%

Table 6: Comparison of Relative Improvement over Parabel**. The relative improvement for each state-of-the-art (SOTA) method is computed based on the metrics reported from its original paper as denoted in the Source column.**

REFERENCES

[1] Rohit Babbar and Bernhard Schölkopf. 2017. DiSMEC: distributed sparse machines for extreme multi-label classification. In *WSDM*.

[2] Rohit Babbar and Bernhard Schölkopf. 2019. Data scarcity, robustness and extreme multi-label classification. *Machine Learning* (2019), 1–23.

[3] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *NIPS*.

[4] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *International Conference on Learning Representations*.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[6] Chuan Guo, Ali Mousavi, Xiang Wu, Daniel N Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. 2019. Breaking the Glass Ceiling for Embedding-Based Classifiers for Large Output Spaces. In *Advances in Neural Information Processing Systems*. 4944–4954.

[7] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. 2019. 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*. ACM, 528–536.

[8] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *KDD*.

[9] Sujay Khandagale, Han Xiao, and Rohit Babbar. 2019. Bonsai-Diverse and Shallow Trees for Extreme Multi-label Classification. *arXiv preprint arXiv:1904.08249* (2019).

[10] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[11] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

[12] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 115–124.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[14] Mikko I Malinen and Pasi Fränti. 2014. Balanced k-means for clustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 32–41.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[16] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing Subset Accuracy with Recurrent Neural Networks in Multilabel Classification. In *NIPS*.

[17] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. 2015. LSHTC: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581* (2015).

[18] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.

[19] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[20] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *WWW*.

[21] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*.

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[23] Sashank J Reddi, Satyen Kale, Felix Yu, Dan Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. 2019. Stochastic Negative Mining for Learning with Large Output Spaces. In *AISTATS*.

[24] Yukihiro Tagami. 2017. 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*. 455–464.

[25] Manik Varma. 2019. The Extreme Classification Repository: Multi-label Datasets & Code. http://manikvarma.org/downloads/XC/XMLRepository.html.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

[27] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[28] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[29] Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. 2018. A no-regret generalization of hierarchical softmax to extreme multi-label classification. In *NIPS*.

[30] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NIPS*.

[31] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. PPDsparse: A parallel primal-dual sparse method for extreme classification. In *KDD*. ACM.

[32] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *Advances in Neural Information Processing Systems*. 5812–5822.