X-BERT: eXtreme Multi-label Text Classification using Bidirectional Encoder Representations from Transformers

Wei-Cheng Chang[1] Hsiang-Fu Yu[2] Kai Zhong[2] Yiming Yang[1] Inderjit Dhillon[2,3]

[1]Carnegie Mellon University, [2]Amazon, [3]University of Texas at Austin

Abstract

Extreme multi-label text classification (XMC) concerns tagging input text with the most relevant labels from an extremely large set. Recently, pretrained language representation models such as BERT (Bidirectional Encoder Representations from Transformers) have been shown to achieve outstanding performance on many NLP tasks including sentence classification with small label sets (typically fewer than thousands). However, there are several challenges in extending BERT to the XMC problem, such as (i) the difficulty of capturing dependencies or correlations among labels, whose features may come from heterogeneous sources, and (ii) the tractability to scale to the extreme label setting because of the Softmax bottleneck scaling linearly with the output space. To overcome these challenges, we propose X-BERT, the first scalable solution to finetune BERT models on the XMC problem. Specifically, X-BERT leverages both the label and input text to build label representations, which induces semantic label clusters to better model label dependencies. At the heart of X-BERT is a procedure to finetune BERT models to capture the contextual relations between input text and the induced label clusters. Finally, an ensemble of the different BERT models trained on heterogeneous label clusters leads to our best final model, which leads to a state-of-the-art XMC method. In particular, on a Wiki dataset with around 0.5 million labels, the precision@1 of X-BERT is 67.87%, a substantial improvement over the neural baseline fastText and a state-of-the-art XMC approach Parabel, which achieve 32.58% and 60.91% precision@1, respectively.

1 Introduction

Extreme multi-label text classification (XMC) aims to tag each given text with the most relevant subset of labels from an enormous label collection, where the number of labels could be in the millions or more. Recently, XMC has attracted considerable attention due to the rapid growth of web-scale data in various industrial applications, such as product categorization for e-commerce [1], Bing's dynamic search advertising [28, 27], and tagging of Wikipedia categories in the PASCAL Large-Scale Hierarchical Text Classification (LSHTC) challenge [25], to name just a few.

XMC poses great computational challenges for developing effective and efficient classifiers owing to the extreme number of instances and labels. Figure 1 illustrates an example of the long-tail label distribution of the Wiki-500K dataset [31]. Much progress has been made to address the challenges of scalability and label sparsity in the XMC problem. While one-vs-all (OVA) approaches [2, 3] often achieve strong performance, they suffer from scalability issues. On the other hand, tree-based methods [28, 14] learn an ensemble of weak but fast classification trees, which however leads to a large model size. Label-partitioning approaches [24, 27, 13] build balanced label trees where only leaf nodes are trained with one-vs-all classifiers, achieving comparable accuracy to OVA approaches but not losing computation speed. Nevertheless, most of the traditional methods for extreme multi-label text classification use bag-of-word (BOW) variants as text representation, ignoring higher order context dependency of words, and thus cannot capture the deeper semantics present in text data.

Deep learning models have been proposed to learn powerful input representations for text classification [17, 15, 19] as well as the XMC problem [20, 37]. Very recently, the Natural Language Processing (NLP) community is witnessing a dramatic paradigm shift towards pretrained deep language representation models, which achieve state-of-the-art across many NLP tasks such as question answering, semantic role labeling, parsing, sentence classification with very few labels, and more. Bidirectional Encoder Representations from Transformers (a.k.a BERT [7]) represent one of the latest developments in this line of work. BERT outperforms its predecessors, ELMo [26] and GPT [29], exceeding state-of-the-art by a wide margin on multiple NLP tasks. Nevertheless, it is challenging to finetune BERT models on the XMC task. The main challenges are the difficulty to capture label

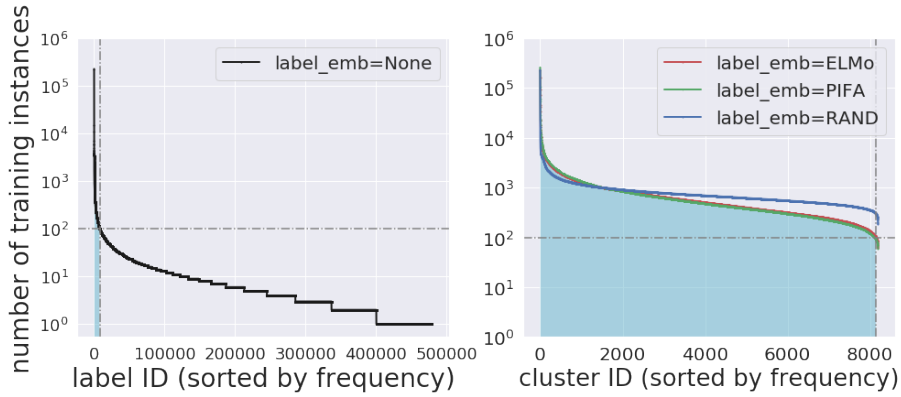

Figure 1: On the left, Wiki-500K shows a long-tail distribution of labels. Note that only 1.6% of the labels have more than 100 training instances, as indicated by the cyan blue regime. On the right is the distribution of the clusters after semantic label indexing, with different label representations. 99.4% of the clusters have more than 100 training instances, which mitigates the sparsity issue for training models in the matching stage.

dependency from heterogeneous sources and the tractability to scale to the extreme-label setting because the additional Softmax layer scales linearly with the large output space.

In this paper, we propose X-BERT, a scalable deep learning approach to finetune BERT-like pre-trained language models for XMC problems. To the best of our knowledge, X-BERT is the first successful finetuned BERT model that outperforms state-of-the-art on the XMC benchmarks that have half a million labels. The contributions of this paper are summarized as follows:

- We propose X-BERT, a scalable BERT finetuning model for XMC problems. X-BERT consists of a Semantic Label Indexing component, a Deep Neural Matching component, and an Ensemble Ranking component.
- We leverage both label text description as well as input keywords to build heterogeneous label representations, which induces semantic label clusters. With the label dependencies encoded in the label clusters, we finetune BERT models to better match the input text to a set of label clusters. Finally, an ensemble of various configurations of X-BERT further improves the performance.
- X-BERT achieves new state-of-the-art results over existing XMC approaches. Quantitatively, on a Wiki dataset with around 0.5 millions of labels, the precision@1 of X-BERT reaches 67.87%, a substantial improvement over the deep learning baseline fastText [15] and competing XMC approach Parabel [27], which achieve 32.58% and 60.91% precision@1, respectively. This amounts to an 11.43% relative improvement over Parabel, which is indeed significant since the recent approach SLICE [13] is reported to lead to a 5.53% relative improvement.
- The dataset, code, and pretrained models are publicly available: https://github.com/OctoberChang/X-BERT.

2 Related Work

2.1 Extreme Multi-label Classification

We categorize XMC algorithms into four categories: one-vs-all approaches, partitioning methods, embedding-based approaches, and deep learning approaches.

One-Vs-All (OVA) approaches. The naive one-vs-all approach treats each label independently as a binary classification problem. OVA approaches [2, 20, 36, 35] have been shown to achieve high accuracies, but they suffer from expensive computation for both training and prediction when the number of labels is very large. Therefore, several techniques have been proposed to speed up the algorithm. PDSparse [36]/PPDSparse [35] introduce primal and dual sparsity to speed up the training as well as prediction. DiSMEC [2], ProXML [3] and PPDSparse [35] explore parallelism and sparsity to speed up the algorithm and reduce the model size. OVA approaches are also

widely used as building blocks for many other approaches. For example, in Parabel [27] and SLICE [13], linear OVA classifiers with a small output domain are used.

Partitioning methods Consider the instance-label matrix, $Y \in \{0,1\}^{N \times L}$, where N is the number of training samples and L is the number of labels. There are two ways to incorporate partitioning. One is partitioning the input space (rows of Y) and the other is partitioning the label space (columns of Y). When Y is very sparse, the input partition only contains a small subset of labels and the label partition only contains a small subset of instances. Furthermore, by applying tree-based approaches on the label partition allows sublinear time prediction with respect to the label size. For example, Parabel [27] 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 [16] relaxed two main constraints in Parabel: 1) allowing multiway instead of binary partitioning of the label space at each intermediate node; 2) removing strict balancing constraints on the partitions. HAXMLNet [38] replaced the sparse bag-of-words features used in Parabel by neural representations from attention networks. SLICE [13] considers building an approximate nearest neighbor (ANN) graph to group the labels. For a given instance, the relevant labels can be quickly found from the nearest neighbors of the instance via the ANN graph.

Embedding-based Approaches Embedding models [4, 39, 5, 6, 12, 33] use a low-rank representation for the label matrix, so that the similarity search for the labels can be performed in a low-dimensional space. In other words, embedding-based methods assume that the label space can be represented by a low-dimensional latent space where similar labels have similar latent representations. To achieve similar computational speedup in practice, nevertheless, embedding-based models often show inferior performance compared to sparse one-vs-all approaches, such as PPD-Sparse [35], and partitioning approaches such as Parabel [27], which may be due to the inefficiency of the label representation structure.

Deep Learning Approaches For text inputs, deep learning representations are expected to better capture the semantic information in the input than bag-of-words features, such as TF-IDF features. XML-CNN [20] employed CNN models for representing text input, while AttentionXML [37] and HAXMLNet [38] used attention models to extract the embeddings from text inputs. SLICE also used supervised pre-trained embeddings from XML-CNN models for training. Recently pre-trained deep language models such as BERT [7], ELMo [26] and GPT [29] have shown promising results on multiple NLP tasks. However, previous work has not been able to incorporate these pre-trained large models for XMC, which presents substantial challenges in both training and inference.

3 Proposed Algorithm: X-BERT

Our approach is partly inspired from the information retrieval (IR) perspective, where the goal is to find relevant documents, from an extremely large set, for a given query. To handle the vast number of documents, an IR engine typically performs search in the following steps [10] — 1) indexing: build an efficient data structure to index the documents; 2) matching: find the document index that this document instance belongs to; 3) ranking: sort the documents in the retrieved index.

An XMC problem can be connected to an IR problem as follows: the large number of labels can be viewed analogously to the large number of documents indexed by a search engine; and the instance to be labeled can be viewed as the query. Due to the success of the three-stage framework of IR for extremely large number of targets, some existing approaches are closely related to this framework, e.g., HAXMLNet and Parabel. In this paper, we propose X-BERT (*BERT for eXtreme Multi-label Text Classification*) under the three-stage framework, which consists of the following stages:

1. *semantically indexing the labels*,
2. *matching the label indices using deep learning*,
3. *ranking the labels from the retrieved indices and taking an ensemble of different configurations from previous steps*.

3.1 Problem Definition

Notation and Definitions Formally, multi-label classification is the task of learning a function f that maps an input (or instance) $\mathbf{x} \in \mathcal{X}$ to its target $\mathbf{y} = [y_1, y_2, \cdots, y_L] \in \mathcal{Y} = \{0,1\}^L$, where L is the number of total unique labels. Assume that we have a set of N training samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, where $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \{0,1\}^L$. We use $Y \in \{0,1\}^{N \times L}$, whose i-th row is $\mathbf{y}_i^\top$, to represent the label matrix. For some special datasets, we have additional label information. For example, each label in the Wikipedia dataset [25] is named by words, such as "Zoos in Mexico" and "Bacon drinks". So we will use $\{\mathbf{z}_j\}_{j=1,2,\cdots,L} \in \mathcal{Z}^L$ as the feature representations of the labels, which may either come from the label information itself or from other approaches.

A Probabilistic Perspective. We formulate our framework for X-BERT in a probabilistic perspective. Assume after indexing, we have K clusters of labels, $\{\mathcal{I}_k\}_{k=1,2,\cdots K}$, where each $\mathcal{I}_k$ is a subset of the label indices, i.e., $\mathcal{I}_k \subset [L]$. For a given instance $\mathbf{x}$, the probability of l-th label y_l being relevant to $\mathbf{x}$ is $P(y_l|\mathbf{x})$. We can form the probabilistic model as follows:

$$P(y_l|\mathbf{x}) = \sum_{k=1}^{K} P(y_l|\mathcal{I}_k, \mathbf{x}, \Theta_r) P(\mathcal{I}_k|\mathbf{x}, \Theta_m). \tag{1}$$

Here $P(\mathcal{I}_k|\mathbf{x}, \Theta_m)$ is the matching model with Θ_m as the parameters and $P(y_l|\mathcal{I}_k, \mathbf{x}, \Theta_r)$ is the ranking model with Θ_r as the parameters. For the ranking model, we impose that

$$P(y_l = 1|\mathcal{I}_k, \mathbf{x}, \Theta_r) = 0, \text{ if } l \notin \mathcal{I}_k,$$

i.e., during the ranking stage, only labels in the retrieved clusters are considered. The intuition is that when the label size is enormous, there will be many similar labels which can be grouped. Our framework has the following advantages:

1. Armed with heterogeneous label representations, we bootstrap the matching and ranking models with various clustering indices $\mathcal{I}$, which leads to more diverse matching/ranking models for a better retrieval system.
2. Driven by the induced yet compact cluster space, the 12 layers of BERT are now a computational feasible option as a realization of the matching model $P(\mathcal{I}_k|\mathbf{x}, \Theta_m)$.
3. Constraining the ranking to a smaller set of labels helps exclude irrelevant labels if the clustering and the matching models are sufficiently good.

We now briefly touch on each of these stages.

3.2 Semantic Label Indexing

Indexing documents in a search engine requires rich text information while the labels of XMC typically lack this information. Thus, we aim to find meaningful label representations in order to build such a semantic indexing system.

Label embedding via label text Instead of using label IDs, we need some semantic information about the labels. Given text information about labels, such as a short description of categories in the Wikipedia dataset, we can use this short text to represent the labels. In this work, we use one of the state-of-the-art word representations, ELMo [26], to represent the words in the label. Note that without finetuning with proper loss, the token embedding of BERT or other variants may not be suitable for the clustering problem. The label embedding is created by a mean pooling over all word embeddings in the label text. In particular, assume the word sequence for the l-th label is $\{w_1, ..., w_k\}$, the label embedding for the l-th label is $\mathbf{z}_l = \frac{1}{k} \sum_{t=1}^{k} \phi_{\text{ELMo}}(w_t)$ where $\phi_{\text{ELMo}}(w_t)$ is the contextual word representation of w_t.

Label embedding via keywords from positive instances However, the short text information for labels may not contain sufficient information and some words in the short text might be ambiguous and noisy. Therefore we consider another label representation derived from the sparse text embedding of instances. Specifically, the label embedding $\mathbf{z}_l$ is the sum of the sparse TF-IDF features of all the relevant instances for the lth label:

$$\mathbf{z}_l = \mathbf{v}_l / \|\mathbf{v}_l\|, \ \mathbf{v}_l = \sum_{i:y_{il}=1} \phi_{\text{TF-IDF}}(\mathbf{x}_i), \quad l = 1, \ldots, L,$$

We refer to this type of label embedding as Positive Instance Feature Aggregation, shorthand as PIFA, which is also used in some state-of-the-art XMC methods [27, 13, 38, 16].

Label indexing With the above label representations, we build the indexing system by clustering the labels as in label partitioning methods [27, 13, 38, 16]. For simplicity, we consider balanced k-means clustering [22, 27] 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 compared to that for an IR problem. 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.

3.3 Deep Neural Matching

The matching phase for XMC is to assign relevant clusters (i.e., indices) to each instance, which is reduced to yet another multi-label classification (MLC) problem. The key to a successful search engine is a high-recall matching model since the subsequent ranking phase is based on the retrieved documents from the matching phase. To build a strong MLC-matching system, we aim to extract the discriminative information in the input text. Many deep learning models have been proposed for the MLC problem such as Seq2Seq [23], CNN [17] and self-attention models [19, 32, 37] to extract the sequential information in the input text. However, deep learning models suffer from high computational complexity and are difficult to scale for XMC. Fortunately, in X-BERT, the number of clusters can be controlled by the practitioner, hence we can govern the scale of MLC-matching problem so that our deep learning models still enjoy reasonable training and inference time.

After label clustering, the labels are partitioned into K clusters $\{\mathcal{I}_k\}_{k=1}^K$, where $\mathcal{I}_k \subset [L]$. The deep neural matching stage aims to find a powerful encoder g to create an instance embedding $\mathbf{u} = g(\mathbf{x})$, and learn shallow neural networks that map the instance embedding $\mathbf{u}$ to the relevant clusters in $\{\mathcal{I}_k\}_{k=1}^K$. Concretely, the cluster $\mathcal{I}_k$ is relevant to an instance $\mathbf{x}_i$ if the instance has a positive label in $\mathcal{I}_k$ (i.e., $y_{il} = 1, \ \exists\, l \in \mathcal{I}_k$).

3.3.1 X-ttention

Deep neural models have demonstrated great success in many NLP applications, such as gated convolution networks for sequence learning [9], self-attention mechanism for text classification [19, 37], as well as Transformer models and its variants for machine translation [32]. Thus, we first consider the self-attention mechanism [19, 32] as a realization for the encoder $g(\cdot)$, hence the name X-ttention.

Specifically, given an instance of T tokens, represented as a sequence of word embeddings $D = \{w_1, \ldots, w_T\}$, we consider BiLSTM [11] to extract higher-order dependencies in text: $H = (\mathbf{h}_1, \ldots, \mathbf{h}_T), \quad \mathbf{h}_t = (\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t)$, where $H \in \mathbb{R}^{2m \times T}$ and $\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t \in \mathbb{R}^m$ are the hidden state of bidirectional LSTM for token t. To have a fixed size embedding for a variable-length instance, X-ttention learns self-attention weights to linearly combine the T hidden states in H. Concretely, self-attention mechanism takes H as input, and outputs a vector of weights $\mathbf{a}$: $\mathbf{a} = \text{softmax}\big(\mathbf{w}_{s2}\tanh(W_{s1}H)\big), \quad \mathbf{a} \in \mathbb{R}^T$. A multi-head attention extends the self-attention by modeling multiple r semantic aspects of the document via weight matrix $W_{s2} \in \mathbb{R}^{r \times d_a}$: $A = \text{softmax}\big(W_{s2}\tanh(W_{s1}H)\big), \quad A \in \mathbb{R}^{r \times T}$. Finally, the instance embedding $\mathbf{u} \in \mathbb{R}^{2rm}$ becomes $\mathbf{u} = g(\mathbf{x}; W_{s2}, W_{s1}, \boldsymbol{\theta}_{\text{BiLSTM}}) = \text{vec}(HA^T)$.

3.3.2 X-BERT

Recently, the NLP community has witnessed a dramatic paradigm shift from task-specific neural architecture to universal pretrained deep language representation models. Under this paradigm, a neural network is first pre-trained on vast amounts of text under an unsupervised objective and then fine-tuned on task-specific data, which achieves state-of-the-art across many natural language understanding tasks such as question answering, semantic role labeling, parsing, sentence classification with very few labels, and more. Bidirectional Encoder Representations

Dataset	N_{trn}	N_{val}	N_{tst}	#features	#labels	#labels/instance	#instances/label	#clusters
Eurlex-4K	13,905	1,544	3,865	33,246	3,714	5.32	19.93	64
Wiki10-28K	11,265	1,251	5,732	99,919	28,139	18.68	7.47	512
AmazonCat-13K	1,067,616	118,623	306,782	161,925	13,234	5.04	406.77	256
Wiki-500K	1,411,760	156,396	676,730	517,631	479,315	4.90	14.44	8192

Table 1: Data Statistics. $N_{trn}, N_{val}, N_{tst}$ refer to the number of instances in training, validation, and test set, respectively.

from Transformers (a.k.a BERT [7]) stands as the latest development in this direction that significantly outperforms many predecessors such as the Generative Pretrained Transformer (GPT) [29] and Embeddings from Language Models (ELMo) [26].

In this paper, we propose to finetune the BERT model as an encoder $g(\cdot)$ for the XMC-matching problem, hence the name X-BERT. To the best of our knowledge, BERT has not yet been explored for the XMC problem that has hundreds of thousands labels or more. Following the setting of [7], we begin with the pre-trained BERT model with 12 layers of Transformer cells and take the final hidden state of the [CLS] token as instance embedding $\mathbf{u} \in \mathbb{R}^m$. During fine-tuning, we optimize the model end-to-end using Adam with a warmup on the learning rate, with an additional linear classifier parameters $W \in \mathbb{R}^{K \times m}$, followed by square hinge loss as the loss function.

While we consider BERT models as an instance of the proposed framework, it is important to note that X-BERT can accommodate other advanced pretrained models such as XLNet [34] and Roberta [21], which we leave for future exploration.

3.4 Ensemble Ranking

a bit confusing, re-read this

After the matching step, a small subset of label clusters is retrieved and the remaining task is to rank the labels in these clusters. As a ranking model, our goal is to model the relevance between the instance and the retrieved labels. Formally, given a label l and an instance $\mathbf{x}$, we want to find a mapping $h(\mathbf{x}, l)$ that maps the instance feature $\mathbf{x}$ and the label l to a score. In this paper, we mainly use the linear one-vs-all (OVA) approach, which is one of the most straightforward and best-performing models. This model treats the assignment of an individual label to an instance as an independent binary classification problem. The class label is positive if the instance belongs to the cluster; otherwise, it is negative. If the instance feature is text, the input to the linear classifier can be the tf-idf feature. With the probability score computed via (1), we further ensemble the scores from different X-BERT models, which are trained on different semantic-aware label clusters by using either ELMo or PIFA embeddings.

4 Empirical Results

4.1 Datasets and Preprocessing

We consider four multi-label text classification datasets from the publicly available Extreme Classification Repository [31] for which we had access to the raw text representation, namely Eurlex-4K, Wiki10-28K, AmazonCat-13K and Wiki-500K. Summary statistics of the datasets are given in Table 1. We follow the training and test split of [31] and set aside 10% of the training instances as the validation set for hyperparameter tuning.

We note that the data statistics and number of labels in Table 1 are slightly different from [31] due of two reasons. First, since only the title of body text is provided in Wiki10-28K and Wiki-500K, we map the title with the latest Wikipedia dump database, and extract the raw text of the document. This creates a subset of the original dataset, yielding slightly smaller number of labels. Second, we adhere to the text preprocessing procedure of [23], replacing numbers with a special token; building the uni-gram vocabulary for TF-IDF; and truncating the documents after 300 words.

We also consider a proprietary E-commerce dataset[1], namely Prod2Query-1M, which maps the product title to relevant customer queries. Prod2Query-1M consists of 14 million instances (products) and 1 million labels (queries) where the label is positive if a product is clicked at least once by the customer's search query corresponding to the label. We divide the dataset into 12.5 million training samples, 0.8 million validation samples and 0.7 million testing samples.

[1] "E-commerce" is a placeholder to keep anonymity during the review period. We will include the details later.

Dataset	Method	Prec@1	Prec@3	Prec@5	Recall@1	Recall@3	Recall@5
Eurlex-4K	PD-Sparse [36]	79.97	66.74	55.50	16.45	40.18	54.66
	fastText [15]	73.97	62.25	51.97	15.07	37.30	51.05
	FastXML [28]	76.17	61.86	50.75	15.54	37.01	49.75
	Parabel [27]	82.48	69.95	58.49	16.87	41.98	57.46
	X-BERT	**86.00**	**74.52**	**62.64**	**17.63**	**44.83**	**61.59**
Wiki10-28K	PD-Sparse [36]	82.12	71.00	60.47	5.04	12.86	18.04
	fastText [15]	65.28	53.48	45.36	3.97	9.62	13.42
	FastXML [28]	83.20	68.68	58.39	5.03	12.28	17.13
	Parabel [27]	82.78	71.70	62.48	5.02	12.88	18.46
	X-BERT	**85.75**	**75.19**	**65.13**	**5.24**	**13.57**	**19.24**
AmazonCat-13K	PD-Sparse [36]	89.18	69.95	55.46	25.44	54.72	67.55
	fastText [15]	81.56	70.65	58.35	22.81	54.36	69.91
	FastXML [28]	92.68	77.17	62.05	26.44	58.70	73.18
	Parabel [27]	91.42	76.34	61.68	25.82	57.84	72.53
	X-BERT	**95.17**	**80.65**	**65.19**	**27.15**	**61.15**	**76.32**
Wiki-500K	PD-Sparse [36]	-	-	-	-	-	-
	fastText [15]	32.58	23.00	18.60	10.67	19.89	25.30
	FastXML [28]	43.46	29.03	22.12	12.30	21.87	26.32
	Parabel [27]	60.91	41.33	31.67	18.74	33.21	39.75
	X-BERT	**67.87**	**46.73**	**35.97**	**21.21**	**37.70**	**45.20**

Table 2: Overall Comparison of X-BERT over state-of-the-art methods. All comparing baselines presented in this Table are re-run on our benchmark train/test set for fair comparison. Note that we defer the discussion of comparing X-BERT to more recent state-of-the-art XMC methods in Table 4.

4.2 Algorithms and Hyperparameters

Comparing Methods. We now compare the proposed X-BERT to state-of-the-art XMC methods including the input partitioning method FastXML [28], the label partitioning method Parabel [27], OVA-based approach PD-Sparse [36], and the representative deep learning model fastText [15] on publicly available benchmark multi-label datasets [31]. Crucially, all evaluation results of the methods in Table 2 are obtained by running their available code on our benchmark dataset partitions. For more comprehensive evaluation with other state-of-the-art approaches that have not released their code or are difficult to reproduce, we have a detailed comparison in Table 4.

Evaluation Metric. We follow [7] to obtain WordPiece tokenized text representation for X-BERT and use TF-IDF unigram for feature-based methods (PD-Sparse, FastXML, and Parabel). 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 [28, 4, 14, 36, 27, 30].

Hyperparameters. For X-BERT, all hyperparameters are chosen from the held-out validation set. The number of clusters are listed in Table 1, which are consistents to Parabel setting for fair comparison. The neural matcher of X-BERT is an uncased BERT_{base} pretrained model configuration, and use Adam [18] as the optimizer with learning rate chosen from $\{5 \times 10^{-5}, 8 \times 10^{-5}, 10^{-4}\}$. The hyperparameters of other methods are set by following their default setting. Specifically, the number of trees in FastXML is $T = 100$, and maximum instances in a leaf node is $m = 10$. For Parabel, the number of trees is $T = 1, 2, 3$, and the maximum number of labels in a leaf node is $m = 100$. Both FastXML and Parabel use $C = 1$ as the loss penalty for the linear SVM with the squared hinge loss, as implemented via LIBLINEAR [8]. For PD-Sparse, the regularization term $\lambda = 0.01$, and the maximum number of iterations is 20. For the deep learning baseline fastText, we set the learning rate to 1, the number of hidden units to 100, and the maximum number of epochs to 1000.

Dataset	Configuration ID	Ablation Configuration				Evaluation Metric					
		indexing	matching	ranking	#trees	Prec@1	Prec@3	Prec@5	Recall@1	Recall@3	Recall@5
AmazonCat-13K	0	ELMo,PIFA	BERT	linear	6	**95.17**	**80.65**	**65.19**	**27.15**	**61.15**	**76.32**
	1	ELMo,PIFA	BERT	linear	2	94.62	80.02	64.51	26.95	60.71	75.61
	2	ELMo	BERT	linear	2	94.49	79.92	64.39	26.89	60.66	75.51
	3	PIFA	BERT	linear	2	94.34	79.67	64.22	26.84	60.42	75.24
	4	ELMo	BERT	linear	1	93.46	78.64	63.06	26.54	59.71	74.08
	5	PIFA	BERT	linear	1	93.50	78.93	63.42	26.54	59.90	74.39
	6	PIFA	BERT	tf-idf	1	56.35	44.43	36.39	15.65	35.05	46.04
	7	ELMo	Xttention	linear	1	92.12	76.75	61.49	26.11	58.36	72.56
	8	PIFA	Xttention	linear	1	92.21	77.03	61.91	26.13	58.58	72.93
	9	ELMo	linear	linear	1	89.81	74.24	59.57	25.27	56.34	70.37
	10	PIFA	linear	linear	1	90.80	75.75	61.05	25.62	57.53	71.95
	11	Random	linear	linear	1	88.91	72.78	58.05	24.99	55.32	68.84
Wiki-500K	0	ELMo,PIFA	BERT	linear	6	**67.87**	**46.73**	**35.97**	**21.21**	**37.70**	**45.20**
	1	ELMo,PIFA	BERT	linear	2	66.29	45.31	34.79	20.63	36.42	43.60
	2	ELMo	BERT	linear	2	65.56	44.66	34.17	20.27	35.76	42.71
	3	PIFA	BERT	linear	2	65.45	44.75	34.38	20.28	35.90	43.01
	4	ELMo	BERT	linear	1	63.74	43.07	32.80	19.61	34.32	40.80
	5	PIFA	BERT	linear	1	64.01	43.48	33.26	19.74	34.74	41.44
	6	PIFA	BERT	tf-idf	1	46.47	27.33	19.79	15.19	23.45	26.70
	7	ELMo	Xttention	linear	1	62.59	42.15	31.95	19.14	33.39	39.53
	8	PIFA	Xttention	linear	1	62.99	42.67	32.63	19.35	34.00	40.60
	9	ELMo	linear	linear	1	54.14	36.70	28.01	16.32	29.22	35.02
	10	PIFA	linear	linear	1	59.21	40.19	30.85	18.12	32.17	38.62
	11	Random	linear	linear	1	40.09	24.61	17.94	11.35	18.99	22.00

Table 3: Ablation study of X-BERT on AmazonCat-13K and Wiki-500K dataset. We outline four take away messages: (1) Index= $\{1, 2, 3\}$ demonstrates a better performance of combining different label embeddings; (2) Index= $\{4, 7, 9\}$ and Index= $\{5, 8, 10\}$ both suggests that, performance-wise, pre-trained deep transformers $>$ BiLSTM+self-attention $>$ linear models (i.e. Parabel); (3) Index=$\{5, 6\}$ manifest the importance and learning rankers instead of unsupervised TF-IDF ranking. (4) Index=$\{10, 11\}$ indicates the importance of semantic label indexing instead of random clustering.

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-BERT	Table 2	**+4.27%**	**+6.53%**	**+7.10%**	X-BERT	Table 2	**+11.43%**	**+13.07%**	**+13.58%**
SLICE	[13, Table 2]	+4.27%	+3.34%	+3.11%	SLICE	[13, Table 2]	+5.53%	+7.02%	+ 7.56%
AttentionXML	[37, Table 2]	+0.91%	+2.09%	+0.86%	HAXMLNet	[38, Table 3]	+2.40%	+5.75%	+ 6.42%
ProXML	[3, Table 5]	+3.86%	+2.90%	+2.43%	ProXML	[3, Table 5]	+2.22%	+0.82%	+ 2.92%
Bonsai	[16, Table 2]	+0.97%	+1.46%	+1.57%	Bonsai	[16, Table 2]	+0.73%	+0.40%	+ 0.52%
PPD-Sparse	[27, Table 2]	+1.92%	+2.93%	+2.92%	PPD-Sparse	[27, Table 2]	+2.39%	+2.33%	+ 2.88%
DiSMEC	[27, Table 2]	+1.73%	+2.90%	+2.80%	DiSMEC	[27, Table 2]	+2.45%	+2.39%	+ 2.98%
PfastreXML	[27, Table 2]	-8.27%	-8.75%	-8.73%	PfastreXML	[27, Table 2]	-13.13%	-18.58%	-20.31%
XML-CNN	[27, Table 2]	-7.14%	-8.59%	-10.64%	XML-CNN	[27, Table 2]	-12.65%	-20.52%	-22.67%
fastText	Table 2	-3.04%	-4.59%	-5.11%	fastText	Table 2	-46.51%	-44.35%	-41.27%
SLEEC	[16, Table 2]	-3.53%	-6.40%	-9.04%	SLEEC	[16, Table 2]	-29.84%	-40.73%	-45.08%

Table 4: 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. The proposed X-BERT outperforms all other SOTA approaches on two commonly used benchmark datasets: Eurlex-4K and Wiki-500K.

4.3 Empirical Results

In this subsection, we first compare the proposed X-BERT approach with state-of-the-art XMC methods, and present a detailed ablation study of X-BERT.

4.3.1 Overall Comparison.

Table 2 compares the proposed X-BERT with other strong XMC baselines on four benchmark datasets. Note again that here we only present the evaluation results of XMC methods that have available code and reproducible performance as reported in their paper when we run their implementation on our benchmark dataset partitions. X-BERT outperforms the SOTA XMC model Parabel on all datasets. It is worth noting that, on the most challenging dataset Wiki-500K, X-BERT improves over Parabel by around 7%/4% absolute improvement for precision@1 and precision@5. This significant gain stems from two novel techniques: the deep neural matcher and the ensemble of various semantic label representations. Compared to the well-known neural baseline fastText, X-BERT achieves far better performance, though at the cost of longer training time.

4.3.2 Results on Prod2Query-1M Dataset.

We applied the X-BERT model to a keyword recommendation system for e-commence ads campaign, where the advertisers bid keywords for their products. The recommended keyword list is shown in a group of 25 per page. So we calculate precision at 25, 50 and 100. We show the relative improvement of X-BERT over the current production method in Table 5.

Metrics	Prec@25	Prec@50	Prec@100
Relative Improvement	+37.16%	+25.25%	+10.26%

Table 5: Relative improvement of X-BERT over the current production method on an e-commence dataset.

4.3.3 Ablation Study.

We carefully conduct an ablation study of X-BERT as shown in Table 3. We decompose the X-BERT framework into three stages: indexing, matching, and ranking, as introduced in Table 3. The configuration Index 0 represents the final best configuration as reported in Table 2. There are four takeaway messages from this ablation study, and we describe them in the following four paragraphs.

Config. id 0 to 3 shows the effect of ensembling different label representations in the indexing stage. The benefit of ensembling models with different random seeds can be seen from configuration id 0 to 1. In addition, from id 1 to 2 and 3, we observe that ensembling using heterogeneous label embeddings is more effective than using the same label embedding of different random seeds, which is the technique used in Parabel models. This confirms that the diversity of semantic label representations helps the neural matcher in capturing different relevant labels.

Next, we analyze how different matching models affect the performance, as shown in configuration id $4, 5, 7, 8, 9, 10$. It is clear that the finetuned BERT (id $4, 5$) is more powerful than the self-attention models (id $7, 8$), and self-attention models are more effective compared to the hierarchical linear models (id $9, 10$) that are used in Parabel.

The importance of training parametric models in the ranking stage is emphasized in configuration id $5, 6$. TF-IDF essentially re-ranks the labels within the retrieved clusters by word matching with the queried documents without learning any model. We see a considerable performance drop from training linear models to the TF-IDF word matching in the re-ranking stage. This finding suggests the importance of learning more powerful parametric models in the ranking stage; in the future we plan to go beyond linear to neural models for ranking.

Finally, we confirm the necessity of label embeddings and clustering algorithms by comparing to a random clustering assignment heuristic, as shown in id $10, 11$. While random assignment produces more evenly-distributed clusters (See Fig. 1), it fails to form meaningful clusters, which poses difficulties in the matching and ranking stages.

4.4 Cross-Paper Comparisons

Many XMC approaches have been recently proposed. While most of them contain an empirical comparison on a few commonly used datasets, such as Eurlex-4K and Wiki-500K, the evaluation metric of the same method on the "same dataset" varies from paper to paper. For example, the precision@1 of DiSMEC is listed as 63.70% in [13, Table 2] and 70.20% in [16, Table 2]. Similarly, the precision@1 of Parabel on Wiki-500K is listed as 59.34% in [13, Table 2], 68.52% in [27, Table 2], while we see 60.91% as seen in Table 2. The inconsistency is likely due

to differences in data preprocessing, data split, or hyper-parameters. Thus, it is difficult to compare metrics directly from different papers.

Here 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. Then for a competing method X with a metric $m(\mathrm{X})$ on a dataset reported in a paper, we can compute the relative improvement over Parabel as follows: $\frac{m(\mathrm{X})-m(\mathsf{Parabel})}{m(\mathsf{Parabel})} \times 100\%$, where $m(\mathsf{Parabel})$ is the metric obtained by Parabel on the same dataset 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 4. From this table, we can clearly observe that X-BERT brings the most significant improvement over Parabel.

5 Conclusions

In this paper, we propose X-BERT, the *first* deep learning approach with finetuned BERT models that achieves state-of-the-art performance in the XMC problem. The novel semantic label indexing stage endows heterogeneous label partitions that bootstrap various BERT models, resulting in a powerful ensemble model for XMC problem. Quantitatively, on the Wiki-500K dataset, the precision@1 is increased from 60.91% to 67.87% when comparing X-BERT to the strong XMC method Parabel. This amounts to a 11.43% relative improvement over Parabel, which is indeed significant compared to the recent state-of-the-art approach SLICE which has 5.53% relative improvement over Parabel.

References

[1] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *WWW*, pages 13–24. ACM, 2013.

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

[3] Rohit Babbar and Bernhard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 2019.

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

[5] Yao-Nan Chen and Hsuan-Tien Lin. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, 2012.

[6] Moustapha M Cisse, Nicolas Usunier, Thierry Artieres, and Patrick Gallinari. Robust bloom filters for large multilabel classification tasks. In *NIPS*, 2013.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[8] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017.

[10] Google. How search works. https://www.google.com/search/howsearchworks/, 2019. Accessed: 2019-1-18.

[11] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[12] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *NIPS*, 2009.

[13] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *WSDM*, 2019.

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

[15] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *EACL*, 2017.

[16] Sujay Khandagale, Han Xiao, and Rohit Babbar. Bonsai-diverse and shallow trees for extreme multi-label classification. *arXiv preprint arXiv:1904.08249*, 2019.

[17] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[19] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[20] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *SIGIR*, 2017.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[22] Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Joint IAPR International Workshops*, 2014.

[23] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NIPS*, 2017.

[24] Alexandru Niculescu-Mizil and Ehsan Abbasnejad. Label filters for large scale multilabel classification. In *AISTATS*, 2017.

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

[26] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[27] 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 *WWW*, 2018.

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

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

[30] Sashank J Reddi, Satyen Kale, Felix Yu, Dan Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. Stochastic negative mining for learning with large output spaces. In *AISTATS*, 2019.

[31] Manik Varma. The extreme classification repository: Multi-label datasets & code, 2018. Accessed: 2018-10-5.

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

[33] Jason Weston, Samy Bengio, and Nicolas Usunier. WSABIE: Scaling up to large vocabulary image annotation. 2011.

[34] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[35] 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 *KDD*, 2017.

[36] Ian EH Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S Dhillon. PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *ICML*, 2016.

[37] Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. AttentionXML: Extreme multi-label text classification with multi-label attention based recurrent neural networks. *arXiv preprint arXiv:1811.01727v1*, 2018.

[38] Ronghui You, Zihan Zhang, Suyang Dai, and Shanfeng Zhu. HAXMLNet: Hierarchical attention network for extreme multi-label text classification. *arXiv preprint arXiv:1904.12578*, 2019.

[39] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, 2014.