X-BERT: eXtreme Multi-label Text Classification with BERT

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) aims to tag each input text with the most relevant labels from an extremely large label set, such as those that arise in product categorization and e-commerce recommendation. Traditional methods using bag-of-words representation find it difficult to capture higher-order dependencies and semantics present in text data. On the other hand, pretrained language representation models such as BERT achieve remarkable state-of-the-art performance across a wide range of NLP tasks including sentence classification among small label sets (typically fewer than thousands). Indeed, there are several challenges in applying BERT to the XMC problem. The main challenges are: (i) the difficulty of capturing dependencies and correlations among labels, whose features may come from heterogeneous sources, and (ii) the tractability to scale to the extreme label setting as the model size can be very large and scale linearly with the size of the output space. To overcome these challenges, we propose X-BERT, the first feasible attempt to finetune BERT models for a scalable solution to the XMC problem. Specifically, X-BERT leverages both the label and document text to build label representations, which induces semantic label clusters in order to better model label dependencies. At the heart of X-BERT is finetuning 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. Empirically, on a Wiki dataset with around 0.5 million labels, X-BERT achieves new state-of-the-art results where the precision@1 reaches 67.80%, a substantial improvement over 32.58%/60.91% of deep learning baseline fastText and competing XMC approach Parabel, respectively. This amounts to a 11.31% relative improvement over Parabel, which is indeed significant since the recent approach SLICE only has 5.53% relative improvement. The datsets, pretrained models and code are available at `https://github.com/OctoberChang/X-BERT`.

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 [2, 5], Bing's dynamic search advertising [35, 34], and tagging of Wikipedia categories in the PASCAL Large-Scale Hierarchical Text Classification (LSHTC) challenge [31], 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. In particular, for the XMC problem, a large fraction of labels are typically tail labels with very few training instances belonging to them. This distribution, also referred to as power law or Zipf's law, is shown in Figure 1 for a benchmark dataset, Wiki-500K from the XMC repository [42]. In this dataset, only ∼125,000 out of 500,000 labels have more than 10 training instances in them. Tail labels exhibit diversity of the label space, and contain informative content not captured by the head or torso labels.

Much progress has been made to address the challenges of scalability and label sparsity in the XMC problem. For example, one-vs-all (OVA) approaches [3, 4] often achieve very competitive performance but suffer from scalability in both the training and prediction phases. As a remedy to reduce computational complexity, sparse structural regularization [48, 47, 49] is introduced to OVA classifiers. On the other hand, Tree-based methods [35, 17, 38] learn an ensemble of weak but fast classification trees, which however leads to a large model size. Label-partitioning approaches[29, 34, 16] build balanced label

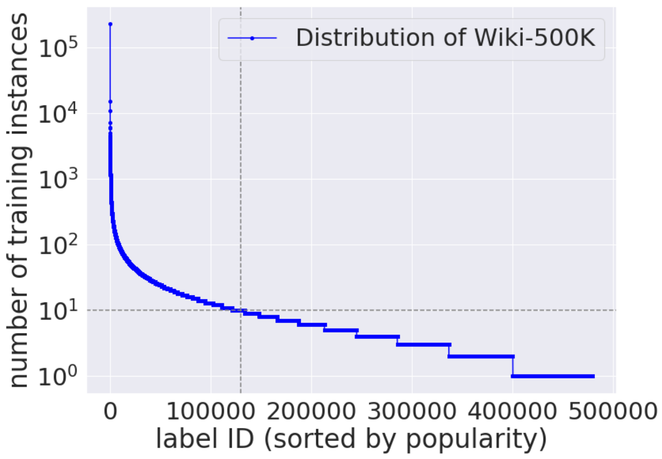

Figure 1: Label frequency in dataset Wiki-500K shows a power-law distribution. X-axis shows the label IDs sorted by their frequency in training instances and Y-axis gives the actual frequency (on log-scale). Note that only 25% of the labels have more than 10 training instances.

trees where only leaf nodes are trained with one-vs-all classifiers, resulting in significant computational gain while achieving comparable accuracy. Nevertheless, most of the traditional methods for XMC use bag-of-word (BOW) variants as text representation, ignoring higher order context dependency of words, and thus cannot capture deeper semantics of text.

Recently, deep learning models have been proposed to learn powerful input representations for text classification [22, 20, 19, 24] as well as the XMC problem [25, 50]. In fact, not until last year, the Natural Language Processing (NLP) community is witnessing a dramatic paradigm shift towards pretrained deep language representation models, which achieves 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 [10]) represents one of the latest developments in this line of work. BERT outperforms its predecessors, ELMo [32] and GPT [36], staggeringly exceeding state-of-the-art by a wide margin on multiple natural language understanding tasks. Nevertheless, it is very challenging to finetune BERT models on XMC task without a careful design. The main challenges are the difficulty to capture label dependency from heterogeneous sources and the tractability to scale to extreme-label setting because of the huge model size and the additional Softmax layer with size depending linearly on the output space.

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

- We propose X-BERT, a scalable BERT finetuning model for extreme multi-label classification 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 document key words to build heterogeneous label representations, which induces semantic-aware 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, ensemble of various configurations of X-BERT further improves the performance.

- In practice, 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 67.80%, a substantial improvement over 32.58%/60.91% of deep learning baseline fastText [20] and competing XMC approach Parabel [34], respectively. his amounts to a 11.31% relative improvement over Parabel, which is indeed significant since the very recent approach SLICE [16] only has 5.53% relative improvement.

2 Related Work

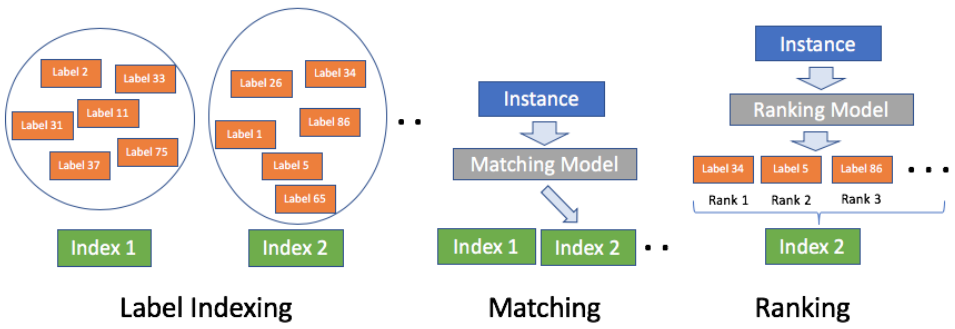

Figure 2: A diagram of the three-stage procedure of X-BERT for extreme multi-label classification

2.1 Extreme Multi-label Classification

We put 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: if the label is relevant to the instance then it is positive; otherwise it is negative. OVA approaches [3, 25, 47, 48] have been shown to achieve high accuracies, but they suffer from expensive computational complexity 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 [48]/PPDSparse [47] introduce primal and dual sparsity to speed up the training as well as prediction. DiSMEC [3], ProXML [4] and PPDSparse [47] 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 [34] and SLICE [16], linear OVA classifiers with a small output domain are used.

Partitioning methods There are two ways to incorporate partitioning: input partitioning [2, 35, 17, 33], label partitioning [34, 18, 41, 45, 29]. Considering 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, input partitioning and label partitioning can be viewed as partitioning the rows and the columns of Y, respectively. When the instance-label matrix is very sparse, for input partitioning, each partition only contains a small subset of labels; for label partitioning, each partition only contains a small subset of instances. Therefore, both partitioning ways can reduce the training and prediction time significantly. Furthermore, most methods, such as [35, 17, 33, 34, 18, 41], apply tree-based approaches, i.e., build a partitioning tree; therefore, a careful choice of tree-based partitioning allows sublinear time prediction with respect to the label size. For example, Parabel [34] partitions the labels through a balanced 2-means label tree using label features constructed from the instances. HOMER [41] and PLT [18] are similar to Parabel. But unlike Parabel, HOMER's label tree is built using the instance-label matrix, while PLT's tree is a probabilistic model. Inspired by Parabel, several approaches are proposed to improve Parabel. Bonsai [21] relaxed two main constraints in Parabel, 1) allowing much bigger partitioning of the label space at each intermediate node instead of two, 2) its partitions do not impose strict balanced-ness constraints. HAXMLNet [51] replaced the sparse bag-of-words features used in Parabel by neural representations from attention networks. SLICE [16] takes a further step for the partitioning. Instead of using fixed tree-based partitioning as in Parabel, SLICE considers building an approximate nearest neighbor (ANN) graph for labels as a special way to group the labels. By properly representing the labels and building the ANN graph, 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 [6, 52, 7, 8, 15, 44] 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. Embedding-based methods explore the relationship among labels through latent subspaces. In other words, embedding-based approaches 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 PD-Sparse [48] / PPD-Sparse [47], and partitioning approaches such as Parabel [34], 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 [25] employed CNN models on the text input, while AttentionXML [50] and HAXMLNet [51] used attention models to extract the embeddings from text inputs. SLICE also used the prefixed embedding from XML-CNN models for training. Recently pre-trained deep language models such as BERT [10], ELMo [32] and GPT [36] have shown promising results on multiple NLP tasks. However, it is still not explored about how to incorporate these pre-trained large models for XMC, which will bring 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 for a given query from an extremely large number of documents. To handle the large number of documents, an IR engine typically performs the search in the following steps [13], 1) indexing: building an efficient data structure to index the documents; 2) matching: finding the document index that this query belongs to; 3) ranking: sorting 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 ensemble of different configurations from previous steps*

See Figure 2 for an illustration of the X-BERT framework.

3.1 Problem Definition

Notation and Definitions Formally, multi-label classification is the task of learning a function f that maps an input $\mathbf{x} \in \mathcal{X}$ 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 [31] 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 assume

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

In other words, during the ranking stage, only labels in the retrieved clusters are considered. This assumption is reasonable for extremely large number of labels because in such a scenario there will be many similar labels and they can be grouped. Under this assumption, our framework has the following advantages:

1. The training time for the ranking model for each cluster can be reduced because it only needs to consider the labels in the cluster and the instances that are relevant to these labels.
2. The prediction time is also reduced, because once a small set of clusters is chosen, we only need to perform ranking for the labels in these clusters.
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 an indexing system. Existing embedding-based XMC approaches [6, 52], for example, only consider the label index and project it to a low-dimensional vector through minimizing the loss between ground-truth labels and predicted labels.

Label embedding via label text Instead of using label IDs, we need some semantic information about the labels. Given some text information about labels, such as a short description of categories in the wikipedia dataset, we can use these short texts to represent the labels. In this work, we use one of the state-of-the-art word representations, ELMo [32], to represent the words in the label. The label embedding is created by a mean pooling over all word embedding 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:

$$\mathbf{z}_l = \frac{1}{k} \sum_{t=1}^{k} \phi_{\mathsf{ELMo}}(w_t)$$

Label embedding via keywords from positive instances However, the short text may not contain sufficient information and some words in the short texts 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:

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

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 [34, 16, 51, 21].

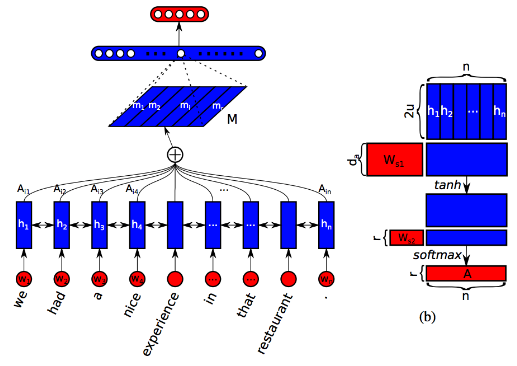

Figure 3: BiLSTM with self-attention. [24]

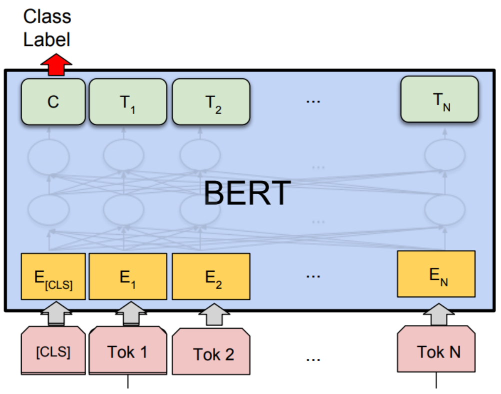

Figure 4: The diagram of BERT models. [10]

Label indexing With the label representations, we build the indexing system by clustering the labels as in label partitioning methods [34, 16, 51, 21]. For simplicity, we consider balanced k-means clustering [27, 34] 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 ranking system 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 exploit the rich semantic information provided by input text documents. Many deep learning models have been proposed for the MLC problem such as Seq2Seq [28], CNN [22, 26] and self-attention models [24, 43, 50] to extract the sequential information in the input text. However, deep learning models suffer from high computational complexity and difficult to scale for XMC. Fortunately, in X-BERT, the number of clusters can be controlled by the practitioner, we can hence govern the scale of MLC-matching problem such that 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 a document embedding $\mathbf{v} = g(\mathbf{x})$, and learn shallow neural networks mapping the document embedding $\mathbf{h}$ 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, 12, 46], self-attention mechanism for text classification [24, 50], as well as Transformer models and its variants for machine translation [43], and more. Thus, we first consider the representative self-attention mechanism [24, 43] as a realization for the encoder $g(\cdot)$, hence the name X-ttention.

Specifically, given a document of T tokens, represented in a sequence of word embeddings $D = \{w_1, \ldots, w_T\}$, we consider BiLSTM [14] to extract higher-order dependency in text:

$$H = (\mathbf{h}_1, \ldots, \mathbf{h}_T), \quad \mathbf{h}_t = (\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t)$$

where $H \in \mathbb{R}^{2u \times T}$ and $\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t \in \mathbb{R}^u$ are the hidden state of bidirectional LSTM for token t. To have a fixed size embedding for a variable length document, 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 document embedding $\mathbf{v} \in \mathbb{R}^{2ru}$ becomes

$$\mathbf{v} = g(\mathbf{x}; W_{s2}, W_{s1}, \boldsymbol{\theta}_{\text{BiLSTM}}) = \text{vec}(HA^T).$$

The procedure is illustrated in Figure 3. Indeed, we will see later in the ablation study experiment (Table 2), the superior performance of self-attention compared to linear models for the XMC problem.

3.3.2 X-BERT

Very recently, the NLP community is witnessing a dramatic paradigm shift from task-specific neural architecture to the 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 from Transformers (a.k.a BERT [10]) stands as the latest developments in this direction that significantly outperforms many predecessors such as the Generative Pretrained Transformer (GPT) [36] and Embeddings from Language Models (ELMo) [32].

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 [10], we begin with the pre-trained BERT model with 12 layers of Transformer blocks and take the final hidden state of the [CLS] input token as document embedding $\mathbf{v} \in \mathbb{R}^u$, as shown in Figure 4. During fine-tuning, we optimize the entire model end-to-end, with the additional linear classifier parameters $W \in \mathbb{R}^{K\times u}$. Specifically, we consider both binary cross entropy or square hinge loss as the loss function, and optimize the model using Adam with a warmup on the learning rate.

Recent studies [40, 39] suggest the pretrained BERT model represents the procedure of traditional NLP pipeline in an interpretable and localizable way such that the regions responsible for each step appear in the expected sequence: POS tagging, parsing, NER, semantic roles, then coreference. Nevertheless, such modeling syntactic structure is arguably [1, 30] less important for document classification compared to complex NLP tasks, such as natural language inference and paraphrasing. It is thus very interesting to examine whether the finetuned X-BERT models still encode sentence structure across a range of syntactic, semantic, local, and long-range dependency, which we leave as future work.

3.4 Ensemble Ranking

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 and an instance, we want to find a mapping $h(\mathbf{x}, l)$ that maps the instance feature $\mathbf{x}$ and the label l into a score. In this paper, we mainly use the linear one-vs-all (OVA) approach. The linear OVA approach is one of the most straightforward and best-performing models. This model treats assigning 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 of the linear classifier can be the tf-idf feature. The output of the classifier is a probability that the instance belongs to the cluster. With the probability score computed via Eq (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 Connections to Existing Work

X-BERT is closely related to label partitioning methods, especially [34, 45, 29]. In the following, we discuss how they are related to X-BERT. [45] has a similar framework as ours during inference. But the order of building the indexing system and learning the matching model is reversed. They assume there is an existing ranking function from a separate training algorithm that maps the instance feature and the label to a score. Their goal is to speed up the prediction without changing the ranking function. To build a faster prediction framework, a partitioner function is first learned such that the instances that are close to each other are mapped to the same partition. Then they assign labels to each partition such that the relevant labels for each instance are in the relevant partition. This framework is different from ours because we consider both training and prediction when building the models. In particular, we first build the indexing system, then learn a matching model from the training data and finally learn a ranking function. [29] applies label filters which pre-select a small set of candidate labels before the base classifier is applied. The label filtering step can be viewed as our matching step. To build the label filters, [29] projects the instance features to a one-dimensional space and learns an upper bound and a lower bound for the range of each label in the one-dimensional space. A label is to be considered for ranking if the projection of the instance falls into the label's range. However, such label partitioning, which is constrained to a one-dimensional space, is limiting for complicated label relationships. In X-BERT, we treat the matching step as a small multi-label classification problem and employ deep learning models to extract the input features. Our framework is also partially inspired by Parabel [34]. However, Parabel mixs up indexing (building the label tree), matching (traversing the internal nodes) and ranking (multi-label classification in the leaf nodes). Our framework separates these three stages and each stage can actually be studied and implemented independently.

As far as we know, X-BERT is the first approach that incorporates pre-trained BERT model into XMC problems. Furthermore, instead of using fixed deep learning embeddings like SLICE, we fine-tune the deep learning models for the specific XMC task.

5 Experiment

Dataset	N_{trn}	N_{val}	N_{tst}	#features	#labels	#labels/instance	#instances/label
Eurlex-4K	13,905	1,544	3,865	33,246	3,714	5.32	19.93
Wiki10-28K	11,265	1,251	5,732	99,919	28,139	18.687	7.47
AmazonCat-13K	1,067,616	118,623	306,782	161,925	13,234	5.04	406.77
Wiki-500K	1,411,760	156,396	676,730	517,631	479,315	4.90	14.44

Table 1: Data Statistics. $N_{trn}, N_{val}, N_{tst}$ refer to the number of instances in training, validation, and test set, respectively. Also note the dataset is denoted by as its name, followed by a dash sign, followed by the number of labels.

5.1 Datasets and Preprocessing

We consider four multi-label text classification datasets downloaded from the publicly available Extreme Classification Repository [42] 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 [42] and set aside 10% of the training instances as the validation set for hyperparameter tuning.

As shown in Table 1, it is important to note that the data statistics, number of labels in particular, are slightly different compared to the Extreme Classification Repository [42] because of two reasons. First, since only the title of body text is provided in Wiki10-28K and Wiki-500K, we map the title with 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 [28], replacing numbers with a special token; building a word vocabulary with the most frequent 80K words; substituting Out-of-vocabulary words with a special token; and truncating the documents after 300 words.

5.2 Algorithms and Hyperparameters

Comparing Methods In Section 5, we compare the proposed X-BERT to state-of-the-art XMC methods including the input partition method FastXML [35], the label partition method Parabel [34], OVA-based approach PD-Sparse [48], and the representative deep learning model fastText [20], on public available benchmark multi-label datasets [42]. Crucially, in this section, all evaluation results of the comparing methods are obtained by running their available code on our benchmark dataset partitions. For more comprehensive evaluation with other state-of-the-art approaches that do not release the code or difficult to reproduce, we have a thorough discussion in Section 6.1.

Evaluation Metric We follow [28] to obtain tokenized text representation for deep learning methods and use TF-IDF unigram features 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 extreme multi-label classification literature [35, 6, 17, 48, 34, 37].

Hyperparameters Setting For X-ttention, the hidden state of the bidirectional LSTM is 512 dimensions in each direction. The self-attention MLP has a hidden layer with $u = 350$ units and set the matrix embedding to have $r = 30$ rows. The final layer is a 2-layer ReLU output MLP with 2000 hidden units. For X-BERT, we consider the uncased BERT$_{base}$ pretrained model configuration that has 12 layers transformer block with 768 hidden units and 12 multiheads attention. We use Adam [23] as the optimizer with learning rate choosing from $\{5 \times 10^{-5}, 8 \times 10^{-5}, 10^{-4}\}$. For X-BERT, we also consider warmup ratio of 0.2.

For the hyperparameter of comparing baselines, we basically follow the default setting as in the papers of these models. Specifically, the number of trees in FastXML is $T = 100$, and maximum instances in 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 L2R L2-loss SVM solver, as implemented via LIBLINEAR [11]. For PD-Sparse, the regularization term $\lambda = 0.01$, and the maximum number of iterations is set to 20 with early stopping via monitoring the Precision@1 on the validation set. 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.

5.3 Empirical Results

In this subsection, we analyze various configurations in the semantic label indexing stage; present the best configuration of our proposed X-BERT approach and compare it with state-of-the-art XMC methods; and investigate different ensemble combinations for ranking stage.

Semantic Labels	Matching Models	matching stage p@1	matching stage p@3	matching stage p@5	ranking stage p@1	ranking stage p@3	ranking stage p@5
ELMo	linear	83.70	71.66	59.43	79.84	67.70	56.30
	X-ttention	87.32	74.91	61.40	83.00	69.81	58.10
	X-BERT	**89.50**	**78.27**	**64.51**	**83.29**	**71.90**	**60.28**
PIFA	linear	91.57	69.12	51.71	81.19	68.86	57.51
	X-ttention	92.91	70.50	52.06	82.82	70.25	58.59
	X-BERT	**93.14**	**74.34**	**55.38**	**82.95**	**70.91**	**59.50**

Table 2: Semantic label indexing on Eurlex-4K dataset.

5.3.1 Analysis on Semantic Label Indexing

Table 2 shows how different label representations and matching models of X-BERT affect the performance of the matching and final ranking stages. For matching algorithms, we compare X-BERT with state-of-the-art deep learning architecture for text classification X-ttention [24, 50] as well as the hierarchical linear models that are used in Parabel [34]. Regarding the semantic label representation, we found that ELMo typically induces better label clusters compared to PIFA for the deep learning based matching models, and leads to the best ranking results.

Finally, we observe the superior performance of X-BERT indeed comes from the improvement of the finetuned BERT models over hierarchical linear models in the matching stage. This empirical results verify our claim that using more complex models, such as deep neural networks, in the matching stage could improve the final ranking performance.

5.3.2 X-BERT Ensemble

Figure 5 illustrates three different ensemble models of 6 configurations of X-BERT. Concretely, X-BERT-v1 ($T = 3$) ensembles 3 BERT models trained on the ELMo label clusters induced by three random seeds. Likewise, X-BERT-v2 ($T = 3$) ensembles 3 BERT models trained on the PIFA label clusters induced by three random seeds, and X-BERT-v3 ($T = 6$) ensembles all combinations (2 label representations $\times$ 3 random seeds). We observe that ensemble using heterogeneous label representation is more effective than ensemble using single label representation of different random seeds, which is the ensemble technique used in Parabel models. This again confirms that the diversity of semantic label representation helps the neural matcher and the final ranking stage. Last but not least, ensembling all 6 configurations yields the state-of-the-art results of X-BERT.

5.3.3 Overall Comparison

Table 3 compares the proposed X-BERT with other strong XMC baselines on four benchmark datasets. Note again that for now 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 state-of-the-art 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, namely the deep neural matcher and the ensemble of various semantic label representations. On the other hand, when compared to fastText, the well-known shallow neural networks for text classifications, X-BERT achieves far better performance though at the cost of longer training time.

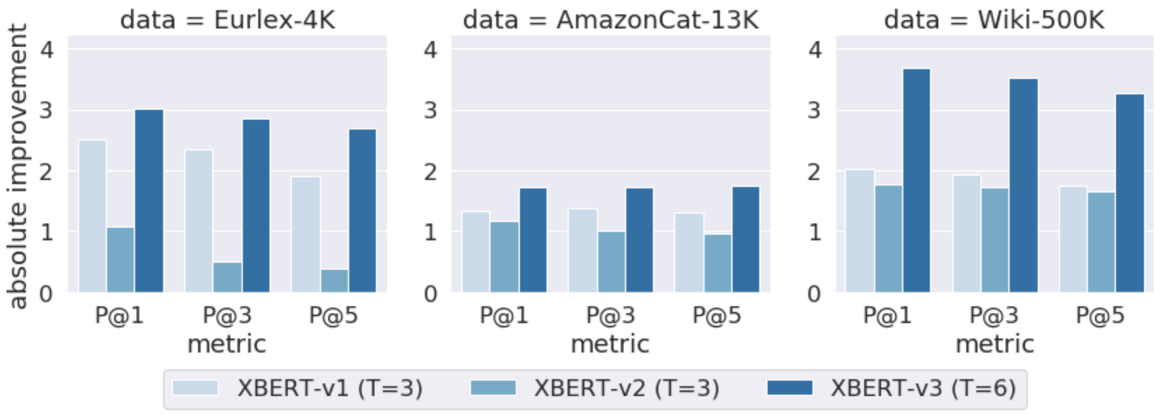

Figure 5: Absolute improvement (%) of various ensemble combinations over single best configuration of X-BERT.

Dataset	Method	Prec@1	Prec@3	Prec@5	Recall@1	Recall@3	Recall@5
Eurlex-4K	PD-Sparse [48]	79.97	66.74	55.50	16.45	40.18	54.66
	fastText [20]	73.97	62.25	51.97	15.07	37.30	51.05
	FastXML [35] (T=100)	76.17	61.86	50.75	15.54	37.01	49.75
	Parabel [34] (T=1)	81.99	68.89	57.30	16.76	41.24	56.26
	Parabel [34] (T=3)	82.48	69.95	58.49	16.87	41.98	57.46
	X-BERT (T=1)	83.29	71.90	60.28	17.00	43.15	59.19
	X-BERT (T=6)	**86.31**	**74.58**	**62.59**	**17.69**	**44.86**	**61.49**
Wiki10-28K	PD-Sparse [48]	82.12	71.00	60.47	5.04	12.86	18.04
	fastText [20]	65.28	53.48	45.36	3.97	9.62	13.42
	FastXML [35] (T=100)	83.20	68.68	58.39	5.03	12.28	17.13
	Parabel [34] (T=1)	82.76	71.37	62.24	5.03	12.82	18.39
	Parabel [34] (T=3)	82.78	71.70	62.48	5.02	12.88	18.46
	X-BERT (T=1)	85.10	73.73	63.30	5.22	13.31	18.70
	X-BERT (T=6)	**85.64**	**75.16**	**65.27**	**5.23**	**13.57**	**19.30**
AmazonCat-13K	PD-Sparse [48]	89.18	69.95	55.46	25.44	54.72	67.55
	fastText [20]	81.56	70.65	58.35	22.81	54.36	69.91
	FastXML [35] (T=100)	92.68	77.17	62.05	26.44	58.70	73.18
	Parabel [34] (T=1)	90.75	75.61	60.99	25.57	57.35	71.80
	Parabel [34] (T=3)	91.42	76.34	61.68	25.82	57.84	72.53
	X-BERT (T=1)	93.27	78.72	63.25	26.47	59.68	74.21
	X-BERT (T=6)	**95.00**	**80.46**	**64.98**	**27.09**	**60.99**	**76.05**
Wiki-500K	PD-Sparse [48]	-	-	-	-	-	-
	fastText [20]	32.58	23.00	18.60	10.67	19.89	25.30
	FastXML [35] (T=100)	43.46	29.03	22.12	12.30	21.87	26.32
	Parabel [34] (T=1)	59.09	39.70	30.25	18.05	31.65	37.64
	Parabel [34] (T=3)	60.91	41.33	31.67	18.74	33.21	39.75
	X-BERT (T=1)	64.12	43.26	32.88	19.75	34.41	40.80
	X-BERT (T=6)	**67.80**	**46.52**	**35.73**	**21.18**	**37.46**	**44.79**

Table 3: Overall Comparison of X-BERT over state-of-the-art XMC methods on benchmark datasets. We consider PD-Sparse, fastText, FastXML and Parabel for comparison because their available implementation codes can successfully reproduce reasonable results. For each method, T in the parentheses denotes the number of single models being ensembled to produce the final best ranking result. Note that PD-Sparse is not scalable to Wiki-500K dataset.

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 3	**4.78%**	**6.46%**	**6.79%**	X-BERT	Table 3	**11.31%**	**12.56%**	**12.82%**
X-ttention	Table 3	2.39%	4.19%	3.74%	X-ttention	Table 3	9.05%	10.16%	10.17%
SLICE	[16, Table 2]	4.27%	3.34%	3.11%	SLICE	[16, Table 2]	5.53%	7.02%	7.56%
AttentionXML	[50, Table 2]	0.91%	2.09%	0.86%	HAXMLNet	[51, Table 3]	2.40%	5.75%	6.42%
ProXML	[4, Table 5]	3.86%	2.90%	2.43%	ProXML	[4, Table 5]	2.22%	0.82%	2.92%
Bonsai	[21, Table 2]	0.97%	1.46%	1.57%	Bonsai	[21, Table 2]	0.73%	0.40%	0.52%
PPD-Sparse	[34, Table 2]	1.92%	2.93%	2.92%	PPD-Sparse	[34, Table 2]	2.39%	2.33%	2.88%
DiSMEC	[34, Table 2]	1.73%	2.90%	2.80%	DiSMEC	[34, Table 2]	2.45%	2.39%	2.98%
PfastreXML	[34, Table 2]	-8.27%	-8.75%	-8.73%	PfastreXML	[34, Table 2]	-13.13%	-18.58%	-20.31%
XML-CNN	[34, Table 2]	-7.14%	-8.59%	-10.64%	XML-CNN	[34, Table 2]	-12.65%	-20.52%	-22.67%
fastText	Table 3	-3.04%	-4.59%	-5.11%	fastText	Table 3	-46.51%	-44.35%	-41.27%
SLEEC	[21, Table 2]	-3.53%	-6.40%	-9.04%	SLEEC	[21, Table 2]	-29.84%	-40.73%	-45.08%

Table 4: Comparison in terms of Relative Improvement over Parabel. The evaluation metric of each state-of-the-art (SOTA) methods are excerpted from the corresponding cited paper, as indicated in the Source column. We then compare the proposed X-BERT model to all other state-of-the-art approaches on the Eurlex-4K and Wiki-500K.

6 Discussions

6.1 Cross-Paper Comparisons

There are many XMC approaches which have been proposed recently. Although 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 data with the same name varies from paper to paper. For example, the precision@1 of DiSMEC is 63.70% from [16, Table 2] and 70.20% from [21, Table 2]. Similarly, the precision@1 of Parabel on Wiki-500K is 59.34% from [16, Table 2], 68.52% from [34, Table 2], and 60.91% from Table 3. These differences can be explained by various data preprocessing, various data split, or various hyper-parameters. Thus, it is not feasible to compare of metrics directly obtained from different papers.

Here we propose an approach to calibrate these numbers such that various methods can be compared in a more principled way. In particular, for each metric $m(\cdot)$, we propose to use the relative improvement over a common anchor method. Given that Parabel is commonly included in many recent papers, we consider Parabel as our anchor method. Then for a competing method X with a number $m(\mathrm{X})$ on a dataset reported in a paper, we can compute the relative improvement over Parabel as follows:

$$\frac{m(\mathrm{X}) - m(\mathsf{Parabel})}{m(\mathsf{Parabel})} \times 100\%, \tag{2}$$

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 into the 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.

6.2 Future Work

End-to-end Training In this paper, the three-stages of X-BERT are performed independently. We believe that if we can perform them all together from end to end, the parameters for each stage can influence each other and achieve better performance. In particular, 1) the label embeddings of the indexing stage are extracted from a fixed pre-trained deep learning model. Can we train it with clustering algorithm as well as the matching model and ranking model? 2) Clustering algorithms in the indexing stage use hard assignments which prevents end-to-end training such as applying the back-propagation algorithm. Can we modify clustering algorithm such that the parameter change becomes continuous and get merged into the overall training. 3) The matching stage uses embeddings extracted from input text using BERT while the ranking model takes the TF-IDF features for the input. Can we replace the TF-IDF features in the ranking stage by the same BERT embeddings in the matching stage while fine-tuning the BERT model from the supervision in both stages?

Latency BERT models are large and expensive to do training and inference. However, many real-world applications deal with huge datasets and need regular model updates. In the meanwhile, some applications demand real-time inference that has latency requirement. Therefore, how to further speed up the training and inference while maintaining model performance is key to applying X-BERT to real-world systems.

Ensemble Methods By using different label representations and different input representations, we are able to obtain different models for X-BERT. Instead of simply taking average of the output results from these models, we can explore more involved ensemble methods.

Dealing with Tail Labels Most XMC problems have a long tailed distribution. We haven't explored much about these tail labels but they play a significant role in the overall performance, especially in real-world applications.

7 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 extreme multi-label text classification problem. The novel semantic label indexing stage endows heterogeneous label partitions that bootstraps the various BERT models, resulting in the powerful ensemble model for XMC problem. Quantitatively, on a Wiki dataset with around 0.5 millions of labels, the precision@1 is increased from 60.91% to 67.80% when comparing X-BERT to the strong XMC method Parabel. This amounts to a 11.31% relative improvement over Parabel, which is indeed significant since the recent state-of-the-art approach SLICE only has 5.53% relative improvement.

References

[1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*, 2019.

[2] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24. ACM, 2013.

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

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

[5] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems*, pages 163–171, 2010.

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

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

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

[9] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, 2017.

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

[11] 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.

[12] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.

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

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

[15] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pages 772–780, 2009.

[16] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 528–536. ACM, 2019.

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

[18] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. Extreme f-measure maximization using sparse probability estimates. In *ICML*, 2016.

[19] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 526–534. JMLR.org, 2016.

[20] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

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

[22] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

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

[24] 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.

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

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *CVPR*, 2015.

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

[28] 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.

[29] Alexandru Niculescu-Mizil and Ehsan Abbasnejad. Label filters for large scale multilabel classification. In *Artificial Intelligence and Statistics*, pages 1448–1457, 2017.

[30] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[31] 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.

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

[33] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 441–449. ACM, 2018.

[34] 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.

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

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[37] 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.

[38] Si Si, Huan Zhang, S Sathiya Keerthi, Dhruv Mahajan, Inderjit S Dhillon, and Cho-Jui Hsieh. Gradient boosted decision trees for high dimensional sparse output. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3182–3190. JMLR. org, 2017.

[39] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[40] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. In *ICLR*, 2019.

[41] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 2008.

[42] Manik Varma. The extreme classification repository: Multi-label datasets & code. `http://manikvarma.org/downloads/XC/XMLRepository.html`, 2018. Accessed: 2018-10-5.

[43] 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.

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

[45] Jason Weston, Ameesh Makadia, and Hector Yee. Label partitioning for sublinear ranking. In *ICML*, 2013.

[46] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *ICLR*, 2019.

[47] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM, 2017.

[48] 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.

[49] Ian En-Hsu Yen, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Sanjiv Kumar, and Pradeep Ravikumar. Loss decomposition for fast learning in large output spaces. In *International Conference on Machine Learning*, pages 5626–5635, 2018.

[50] 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. In *AAAI*, 2019.

[51] 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.

[52] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *International conference on machine learning*, pages 593–601, 2014.