Correlation Networks for Extreme Multi-label Text Classification

Guangxu Xun, Kishlay Jha, Jianhui Sun, Aidong Zhang
Department of Computer Science, University of Virginia, Charlottesville, VA, USA
{gx5bt, kj6ww, js9gu, aidong}@virginia.edu

ABSTRACT

This paper develops the Correlation Networks (CorNet) architecture for the extreme multi-label text classification (XMTC) task, where the objective is to tag an input text sequence with the most relevant subset of labels from an extremely large label set. XMTC can be found in many real-world applications, such as document tagging and product annotation. Recently, deep learning models have achieved outstanding performances in XMTC tasks. However, these deep XMTC models ignore the useful correlation information among different labels. CorNet addresses this limitation by adding an extra CorNet module at the prediction layer of a deep model, which is able to learn label correlations, enhance raw label predictions with correlation knowledge and output augmented label predictions. We show that CorNet can be easily integrated with deep XMTC models and generalize effectively across different datasets. We further demonstrate that CorNet can bring significant improvements over the existing deep XMTC models in terms of both performance and convergence rate. The models and datasets are available at https://github.com/XunGuangxu/CorNet.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; • **Computing methodologies** → **Natural language processing**.

KEYWORDS

multi-label text classification; deep learning; label correlation

ACM Reference Format:
Guangxu Xun, Kishlay Jha, Jianhui Sun, Aidong Zhang. 2020. Correlation Networks for Extreme Multi-label Text Classification. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403151

1 INTRODUCTION

Extreme multi-label text classification (XMTC) is a Natural Language Processing (NLP) task, which aims to tag a given text with the most relevant subset of labels from an extremely large label set. For example, there are more than one million product labels on Amazon and when a new product becomes online, XMTC tries to tag the new product with the most relevant Amazon product labels based on its description text. It is worth mentioning that multi-label classification is different from multi-class classification in that multi-label classification predicts a subset of relevant labels, whereas multi-class classification predicts a single label. Due to the rapid growth of data scale in various industrial applications in recent years, such as product labeling for e-commerce [1], biomedical text annotation [31] and Wikipedia category tagging [25], XMTC has garnered increasing attention.

XMTC is a challenging NLP task as one needs to deal with a huge number of labels and training samples. Current XMTC models can be roughly grouped into four categories: 1) one-vs-all models [2, 37] which learn a separate classifier for each label, 2) embedding based models [5, 30] which represent labels in a low-dimensional embedding space, 3) tree based [15, 16, 18, 27] models which learn a label hierarchy to improve model efficiency and 4) deep learning based models [6, 17, 21, 33, 38] which employ deep learning techniques.

As in many other NLP tasks, deep learning based models have also achieved the state-of-the-art performance in XMTC, thanks to the recent development of deep learning techniques [3, 7, 9, 12, 13, 20, 32]. A deep multi-label text classification model is normally composed of two components: the first component extracts information from the text sequence and the second component converts the extracted features into label predictions. According to the sequence modeling style in the first component, deep XMTC models fall into three main categories: 1) Recurrent Neural Networks (RNN) based [17, 33, 38], 2) Convolutional Neural Networks (CNN) based [21] and 3) Bidirectional Encoder Representations from Transformers (BERT) based models [6]. In contrast to the various choices of sequence modeling styles for the first component, the choices for the second component to predict labels are quite limited and the most common choice is a fully connected (FC) layer. A less popular choice [36] is to follow the sequence-to-sequence (Seq2Seq) [29] paradigm and use an RNN decoder to generate predicted labels sequentially, but the underlying assumption of Seq2Seq is not suitable for XMTC since labels are not ordered.

Using a fully connected layer to map feature vectors to label predictions is simple and straightforward, but it fails to take full advantage of the correlations among different labels. For instance, if we have high confidence that an Amazon product should be labeled with "Blues" and "R&B", then it probably should also be labeled with "Music". This correlation information is able to help us obtain more accurate label predictions. Therefore, in order to make better use of the label correlations, we propose a new network architecture, named Correlation Networks (CorNet). The proposed architecture works as an add-on enhancer module to existing deep XMTC architectures to form a new end-to-end model. The resulting CorNet

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403151

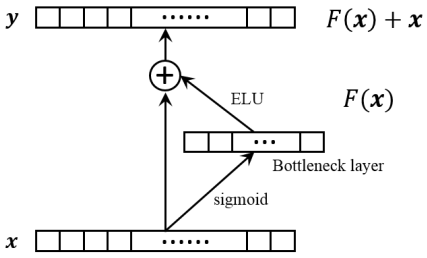


Figure 1: A CorNet building block.

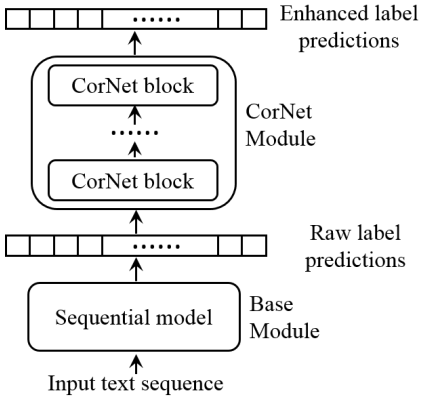


Figure 2: Framework of a CorNet model.

model is able to improve over the original architecture by allowing it to promote correlated predictions and demote uncorrelated predictions. Specifically, CorNet is an independent module that can be added after the last label prediction layer of a deep XMTC model. As a result, a CorNet model capable of utilizing label correlations is constructed. It is noteworthy that, rather than come from some external knowledge base, the label correlation knowledge comes from the label co-occurrence patterns within the XMTC dataset. In other words, CorNet models do not pose more training restrictions or require extra knowledge than regular deep XMTC models. To sum up, CorNet has the following advantages:

- CorNet is able to exploit the correlation information among different labels.
- CorNet is a general and independent architecture that can be directly integrated with any deep XMTC model without changing the model.
- CorNet models consistently achieve significant improvements over the state-of-the-art deep XMTC models on benchmark datasets.
- CorNet models converge faster than the original models during training.

2 RELATED WORK

One-vs-all XMTC models [2, 37] treat each label as an independent binary classification problem. These models usually achieve high accuracy, but inevitably suffer from expensive computational cost, especially when the label set is huge. Tree based XMTC models [15, 16, 18, 27] reduce the computational cost by creating a tree hierarchy for the labels. For example, Parabel [27] recursively partitions the labels into two balanced tree branches and the leaf models are one-vs-all models with much smaller label vocabularies. Embedding based XMTC models [5, 30] use embeddings to represent labels and perform the similarity search for labels in the low-dimensional embedding space. The low-dimensional label embeddings are able to boost the model efficiency, but also hurt the model performance.

The aforementioned approaches represent the traditional XMTC models. In recent years, as deep learning continues to advance in various domains, such as NLP [3, 20, 32, 34, 35] and computer vision [11, 12], deep XMTC models have also gained increasing popularity and achieved better performances than the traditional models [21, 38]. The focus of this paper would also be on deep XMTC models. Unlike most traditional XMTC models taking as input bag-of-words features, e.g., tf-idf features, deep XMTC models take text sequences as input, which are supposed to contain richer semantics as the word order information is preserved. Therefore, the first step of deep XMTC models is to convert an arbitrary-length text sequence to a fixed-dimensional feature vector. AttentionXML [38] and MeSHProbeNet [33] utilize RNNs and attention mechanism to extract information from the input text sequence. The attention mechanism selectively pays attention to different parts of the input based on their semantics and generates a weighted sum vector as the summary of the input text. XML-CNN [21] utilizes CNNs to extract features from the input text sequence. Each CNN filter carries one semantic pattern and outputs a feature map. Then a max pooling operation is applied over the feature map along the sequence dimension to get the feature corresponding to this filter. X-BERT [6] utilizes a BERT encoder with multi-head self-attention to extract information from the input text sequence.

There is one limitation in the existing deep XMTC models, which is the inability to fully exploit the label correlation information. Previous studies [10, 24] have shown that correlations between label pairs contain useful information for label prediction. This inspires us to incorporate label correlations into deep XMTC models. Instead of just using pairwise label correlations, our proposed CorNet architecture is able to exploit label correlations of all patterns.

3 CORNET

A CorNet block is a computational unit which maps raw label predictions to enhanced label predictions based on label correlations. A CorNet building block is illustrated in Figure 1. Formally, a CorNet building block is defined as:

$$\boldsymbol{y} = F(\boldsymbol{x}) + \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ are the input and the output of this CorNet block and F stands for the underlying mapping function. Specifically, $\boldsymbol{x}$ denotes the raw label predictions before the CorNet block and $\boldsymbol{y}$ denotes the enhanced label predictions after the CorNet block. For a regular deep XMTC model without CorNet blocks, output label prediction $\boldsymbol{x}$ is directly sent to the loss function for training or to the sigmoid function for prediction, whereas for a CorNet enhanced deep model,

correlation enhanced label prediction $\boldsymbol{y}$ is used as the final output. We also add an identity mapping between raw predictions $\boldsymbol{x}$ and correlation enhanced predictions $\boldsymbol{y}$. Although theoretically adding identity mappings between layers does not increase the expressive power of a neural network, it is able to address the degradation problem during optimization [12]. The degradation problem refers to the phenomenon that with the network depth increasing, training error unexpectedly gets saturated and then even degrades rapidly. Therefore, it is important for CorNet blocks to have identity mappings to avoid the degradation problem, as the depth of a complete CorNet model has already accumulated in the course of converting text sequences to raw label predictions $\boldsymbol{x}$.

Specifically, F is the correlation enhancing function to be learned. The most straightforward design for function F is one fully connected layer: $F(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}$, where $\boldsymbol{W}$ denotes the weight matrix of the layer, and the bias term and the activation function are omitted for simplifying notations. In this way, the i^{th} enhanced prediction y_i is a linear combination of all raw predictions $\{x_1, x_2, ..., x_i, ...\}$ and hence all possible linear correlations between the i^{th} label and other labels are taken into consideration. However, this design of function F is not practical in terms of memory usage. Let V be the number of distinct labels, then $\boldsymbol{W}$ would be a V-by-V matrix. Since V is usually huge in XMTC tasks, it is hard for $\boldsymbol{W}$ to fit in the GPU memory. In addition, this design would also be a waste of the expressive power of the network, as most labels are uncorrelated to each other and hence most elements in $\boldsymbol{W}$ would be 0s. Moreover, one fully connected layer only allows us to exploit linear label correlations. To this end, we insert a bottleneck layer between $\boldsymbol{x}$ and $\boldsymbol{y}$, as shown in Figure 1. Let R denote the dimension of the bottleneck layer. By having a bottleneck layer and setting $R \ll V$. the model size is significantly reduced and more complex correlations can be captured by the additional layer. Therefore, our design for function F can be formally defined as:

$$F(\boldsymbol{x}) = \boldsymbol{W}_2\delta(\boldsymbol{W}_1\sigma(\boldsymbol{x}) + \boldsymbol{b}_1) + \boldsymbol{b}_2, \tag{2}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2$ are the weight matrices, $\boldsymbol{b}_1, \boldsymbol{b}_2$ are the biases, and σ, δ are the sigmoid activation function and the ELU [8] activation function respectively. Recall that $\boldsymbol{x}$ are the raw label prediction logits, hence we first need to use the sigmoid activation to convert label logits $\boldsymbol{x}$ to label probabilities $\sigma(\boldsymbol{x})$ ranging from 0 to 1, representing the confidence level of each label prediction. The correlations are then calculated based on the label probabilities. By doing so, the interpretability of label correlations is also achieved.

We have assumed that $\boldsymbol{x}$ is the output label predictions of a deep XMTC network, but actually $\boldsymbol{x}$ could also be the output label predictions of a previous CorNet block. This means we can stack any number of CorNet blocks to form a deep CorNet module and the output of each block is a correlation enhancement over the output of the previous block. So more complicated label correlations can be captured by the deep CorNet module. CorNet is a general architecture and it can be concatenated after any standard deep XMTC architectures. Figure 2 illustrates the framework of a CorNet enhanced model. The base module could be seen as a black box which is responsible for converting a text sequence into raw label predictions. The CorNet module then enhances the raw predictions with label correlations and outputs the enhanced label predictions.

4 CORNET INSTANTIATIONS

Depending on how raw label prediction $\boldsymbol{x}$ is derived, CorNet models can have different structures. This flexibility of CorNet allows it to employ different sequence modeling styles as the base module. To illustrate this point, we develop CorNet models by integrating CorNet modules with several standard deep XMTC architectures, described below.

4.1 CorNetXML-CNN

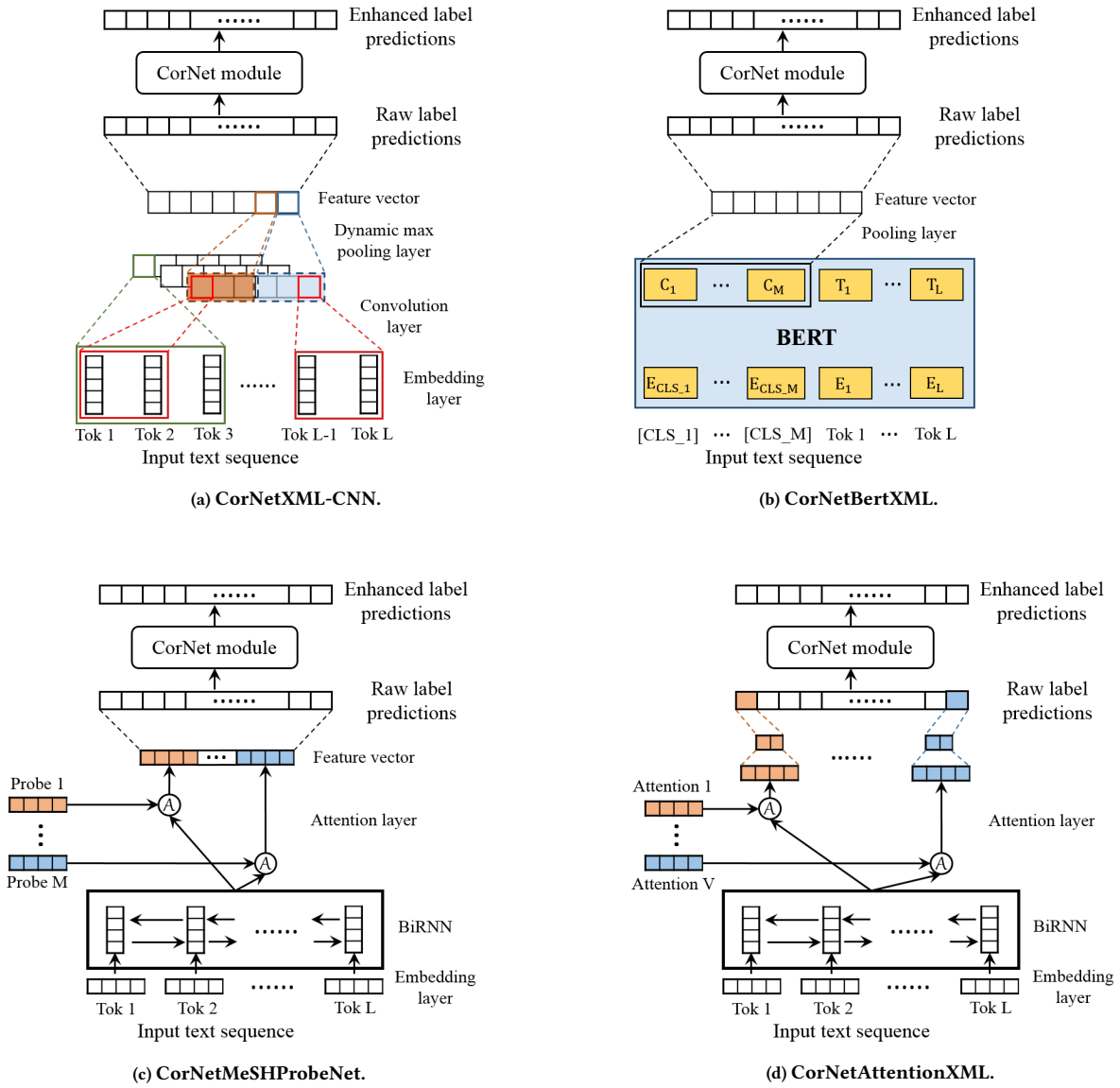
We first consider the construction of CorNet model based on CNN sequence modeling, which we name CorNetXML-CNN, as shown in Figure 3a. The base module is the XML-CNN model [21], where a fixed-dimensional feature vector can be achieved by applying a set of 1D convolution filters and a dynamic max pooling on the input word embeddings. This fixed-dimensional feature vector is used as the aggregate sequence representation. The dimension of the feature vector is proportional to the number of the convolution filters: the more the filters, the more the extracted features. One can adjust the number of convolution filters based on the size of features needed.

4.2 CorNetBertXML

When utilizing BERT [9] in the base module, we need to make some modifications so that it is more suitable for the XMTC task. For classification tasks, BERT always adds a special symbol [CLS] in front of every input sequence and uses the final hidden state corresponding to this token as the aggregate sequence representation. This schema works well for regular classification tasks, but for extreme classification tasks with a huge number of labels, the feature size might be insufficient. For example, the typical embedding dimension for deep XMTC models is 300, then BERT will also output a 300-dimensional hidden state for token [CLS]. Assume there are 300,000 distinct labels, then a 300-dimensional feature vector is probably not informative enough to generate a 300,000-dimensional label prediction. Increasing the feature dimension will drastically increase the model size and reduce the model efficiency, as the embeddings, the hidden states and the intermediate self-attention results will also grow larger. Therefore, we propose to have multiple special tokens to accommodate to extreme classification tasks, as shown in Figure 3b. The multi-head self-attention mechanism of BERT allows each special token to go over the entire input sequence and the final hidden state of each special token can be deemed as a particular feature vector. Specifically, we can add M classification symbols [CLS_1], [CLS_2], ... , [CLS_M] in front of every input sequence and use the concatenation of their final hidden states as the aggregate sequence representation. The cost of using multiple [CLS] tokens to get wider feature vectors is much smaller than increasing the feature dimension. We name this XMTC oriented model BertXML. Accordingly, its CorNet enhanced version is named CorNetBertXML.

4.3 CorNetMeSHProbeNet

MeSHProbeNet [33] was originally proposed for biomedical document annotation, i.e., tagging biomedical documents with relevant Medical Subject Headings (MeSH) terms. MeSH labeling is also



(a) CorNetXML-CNN.

(b) CorNetBertXML.

(c) CorNetMeSHProbeNet.

(d) CorNetAttentionXML.

Figure 3: CorNet instantiations.

a general XMTC task except that it contains extra journal information. Therefore, we first need to remove journal related components from MeSHProbeNet to make it suitable for the general XMTC tasks. MeSHProbeNet models text sequences with bidirectional RNNs. Each MeSH probe is essentially a self-attention that can extract related information from the RNN hidden states and output a fixed-dimensional feature vector. The number of probes M is a preset hyper-parameter. The concatenated feature vector from all probes is used as the aggregate sequence representation. Label prediction $\boldsymbol{x}$ can then be calculated based on the concatenated feature vector. Similarly, we can integrate a deep CorNet module into MeSHProbeNet to enhance the label prediction, as shown in Figure 3c, where Ⓐ denotes attention calculation. We name the new model CorNetMeSHProbeNet.

4.4 CorNetAttentionXML

Although AttentionXML [38] is also built upon bidirectional RNNs and attentions, it is quite different from MeSHProbeNet. In fact, AttentionXML is different from all three basic models we just mentioned above, in the sense that all other models try to extract a feature vector as the aggregate sequence representation for the entire sequence, while AttentionXML extracts a feature vector for every distinct label. Specifically, AttentionXML has a self-attention for every label and each attention generates a feature vector to

Table 1: Dataset statistics. N_{train} and N_{test} refer to the number of instances in training and test set. D is the vocabulary size of the input text. L is the number of distinct labels. $\overline{L}$ and $\widetilde{L}$ denote the average number of labels per instance and the average number of instances per label. $\overline{W}_{train}$ and $\overline{W}_{test}$ refer to the average number of words per training and test instance.

Dataset	N_{train}	N_{test}	D	L	$\overline{L}$	$\widetilde{L}$	$\overline{W}_{train}$	$\overline{W}_{test}$
EUR-Lex	15,449	3,865	186,104	3,956	5.30	20.79	1248.58	1230.40
AmazonCat-13K	1,186,239	306,782	203,882	13,330	5.04	448.57	246.61	245.98
Wiki-500K	1,779,881	769,421	2,381,304	501,008	4.75	16.86	808.66	808.56

predict the corresponding label. Thus, AttentionXML is able to extract specific features for each label, but inevitably suffers from expensive computational cost. For example, the number of attentive probes in MeSHProbeNet is normally $M \in [5, 25]$, while the number of self-attentions in AttentionXML is equivalent to the size of the label set V, which could be in the millions. The final multi-label prediction vector is obtained by concatenating all individual label predictions.

Unlike other models where label prediction $\boldsymbol{x}$ is obtained through a transformation of the aggregate sequence representation, AttentionXML obtains label prediction $\boldsymbol{x}$ through the concatenation of individual predictions. We can still directly add a deep CorNet module onto AttentionXML to form a correlation enhanced model, named CorNetAttentionXML, as shown in Figure 3d. This shows the flexibility and generality of CorNet, as it can be integrated with any standard deep XMTC model regardless of how raw label prediction $\boldsymbol{x}$ is achieved.

5 EXPERIMENTS

In this section, we investigate the effectiveness of CorNet across different datasets and model architectures.

5.1 Datasets and Experimental Settings

We use three benchmark datasets, including one small-scale dataset EUR-Lex [23], one medium-scale dataset AmazonCat-13K [22], and one large-scale dataset Wiki-500K[1]. The dataset statistics are summarized in Table 1. The training and test split is the same as in the data source[1].

We adhere to the text preprocessing procedure of [38]: for each dataset, the vocabulary size is limited to 500,000 words according to the word frequency in the training set. Word embeddings are initialized with the 300-dimensional pretrained GloVe [26] embeddings. Word embeddings are frozen for the EUR-Lex dataset. Input text sequences are truncated to 500 words if longer. All models are trained by the Adam optimizer [19] with a learning rate of 1e-3 on Nvidia Titan V GPUs. Stochastic weight averaging [14] is also utilized to improve generalization.

5.2 Model Configurations

The deep XMTC models included in our experiments are XML-CNN, BertXML, MeSHProbeNet, AttentionXML and their CorNet enhanced counterparts: CorNetXML-CNN, CorNetBertXML, CorNetMeSHProbeNet, CorNetAttentionXML. In order to make the performance comparison as fair as possible, we assign the same embedding dimension and hidden size for all models. The detailed parameter settings for each model are reported below.

5.2.1 XML-CNN. Dimension of embeddings: 300; convolution filter sizes: 2, 4, 8; number of filters of each size: 128; output size of dynamic max pooling: 8; dimension of bottleneck layer: 512; dropout rate: 0.5.

5.2.2 BertXML. Dimension of embeddings: 300; hidden size: 300; number of BERT layers: 3; number of attention heads: 2; intermediate layer size: 600; number of [CLS] symbols: 5; dropout rate: 0.1. For the large dataset (Wiki-500K), we insert a 512-dimensional bottleneck layer before the output layer to improve efficiency.

5.2.3 MeSHProbeNet. Dimension of embeddings: 300; hidden size: 300; number of RNN layers: 2; number of probes: 5; dropout rate: 0.5. For the large dataset (Wiki-500K), we insert a 512-dimensional bottleneck layer before the output layer to improve efficiency.

5.2.4 AttentionXML. For the small dataset (EUR-Lex), we set dimension of embeddings: 300; hidden size: 256; number of RNN layers: 1; dimension of fully connected layers: 256; dropout rate: 0.5. For the medium dataset (AmazonCat-13K), we set dimension of embeddings: 300; hidden size: 200; number of RNN layers: 1; dimension of fully connected layers: 100; dropout rate: 0.5. For the large dataset (Wiki-500K), we are not able to fit the model into our GPU memory, as the model was trained parallelly on 8 GPUs according to the original paper [38].

5.2.5 CorNetXML-CNN. Same settings as XML-CNN, except for the integration of a CorNet module with 2 CorNet blocks.

5.2.6 CorNetBertXML. Same settings as BertXML, except for the integration of a CorNet module with 2 CorNet blocks.

5.2.7 CorNetMeSHProbeNet. Same settings as MeSHProbeNet, except for the integration of a CorNet module with 2 CorNet blocks.

5.2.8 CorNetAttentionXML. Same settings as AttentionXML, except for the integration of a CorNet module with 2 CorNet blocks.

5.3 Evaluation Metrics

Considering the sparsity of labels in XMTC tasks, a short ranked list of potentially relevant labels for each test instance is commonly used to represent classification quality. Following the convention of the XMTC literature [15, 21, 28, 38], we adopt two instance-based ranking metrics to evaluate the models: the precision at top k (precision@k) and the normalized Discounted Cumulative Gain at top k (nDCG@k). Let $\boldsymbol{z} \in \{0, 1\}^L$ denote the ground truth label vector of an instance and $\hat{\boldsymbol{z}} \in \mathbb{R}^L$ denote the model predicted score

[1] http://manikvarma.org/downloads/XC/XMLRepository.html

Table 2: Comparison results on EUR-Lex.

Model	P@1	P@3	P@5	N@1	N@3	N@5	#GPUs	#hours	model size (GB)
XML-CNN	76.81	62.79	51.56	76.81	66.44	60.47	1	0.08	0.22
CorNetXML-CNN	78.60	64.22	53.07	78.60	67.81	61.90	1	0.08	0.28
BertXML	77.80	64.57	53.25	77.80	67.97	62.10	1	0.25	0.24
CorNetBertXML	79.02	65.49	53.94	79.02	68.98	62.97	1	0.29	0.30
MeSHProbeNet	79.92	66.52	55.13	79.92	69.98	64.13	1	1.08	0.26
CorNetMeSHProbeNet	83.47	70.50	58.73	83.47	73.86	67.95	1	1.09	0.32
AttentionXML	85.43	73.30	60.99	85.43	76.54	70.45	1	0.95	0.21
CorNetAttentionXML	86.39	73.70	61.72	86.39	77.10	71.24	1	0.96	0.27

Table 3: Comparison results on AmazonCat-13K.

Model	P@1	P@3	P@5	N@1	N@3	N@5	#GPUs	#hours	model size (GB)
XML-CNN	94.53	79.12	63.38	94.53	88.19	85.61	1	2.57	0.64
CorNetXML-CNN	95.36	80.55	64.83	95.36	89.54	87.11	1	4.17	0.86
BertXML	94.78	80.78	65.51	94.78	89.57	87.58	1	3.88	0.70
CorNetBertXML	95.22	81.37	66.03	95.22	90.13	88.13	1	3.70	0.92
MeSHProbeNet	95.63	81.84	66.47	95.63	90.65	88.69	1	9.08	0.77
CorNetMeSHProbeNet	96.10	82.82	67.57	96.04	91.54	89.78	1	9.22	0.99
AttentionXML	95.13	81.12	66.10	95.13	89.90	88.13	4	14.73	0.62
CorNetAttentionXML	96.19	82.81	67.63	96.19	91.54	89.81	4	20.21	0.84

Table 4: Comparison results on Wiki-500K.

Model	P@1	P@3	P@5	N@1	N@3	N@5	#GPUs	#hours	model size (GB)
XML-CNN	63.36	43.92	33.86	63.36	53.78	51.46	2	19.57	1.63
CorNetXML-CNN	66.26	46.59	35.80	66.26	56.86	54.39	2	22.88	2.44
BertXML	66.86	47.67	36.30	66.86	58.13	55.40	2	30.73	1.64
CorNetBertXML	67.14	47.85	36.43	67.14	58.33	55.59	2	35.53	2.44
MeSHProbeNet	64.57	44.34	34.14	64.57	54.72	52.53	2	40.18	1.65
CorNetMeSHProbeNet	69.79	50.68	38.98	69.79	61.47	58.91	2	43.67	2.45

vector for the same instance, then precision@k and nDCG@k are defined as:

$$\begin{aligned} \text{precision@k} &= \frac{1}{k} \sum_{l \in r_k(\hat{z})} z_l, \\ \text{DCG@k} &= \sum_{l \in r_k(\hat{z})} \frac{z_l}{\log(l+1)}, \\ \text{nDCG@k} &= \frac{\text{DCG@k}}{\sum_{l=1}^{\min(k, \|z\|_0)} \frac{1}{\log(l+1)}}, \end{aligned} \tag{3}$$

where $r_k(\hat{z})$ is the ground truth indices corresponding to the top k indices of the model predicted rank list, and $\|z\|_0$ counts the number of ground truth labels for this instance. Precision@k and nDCG@k are computed for each instance and then get averaged over all instances.

5.4 Performance Comparison

The performance and training details of all models on three datasets are summarized in Tables 2, 3 and 4. Precision@k and nDCG@k, denoted by P@k and N@k for short in the tables, are calculated with k = 1,3,5. Some training details such as the training time and the number of GPUs used are also reported. Due to the GPU memory limitation, experiments for AttentionXML and CorNetAttentionXML are only conducted on EUR-Lex and AmazonCat-13K. It is worth mentioning that all the models reported here are single models, not ensemble models.

As we can observe from Tables 2, 3 and 4, CorNet is able to consistently improve the performance of all deep XMTC models on all datasets in all metrics. Among all the deep models, the improvement over BertXML is the smallest, and we speculate that it is due to the multi-head self-attention mechanism and the point-wise

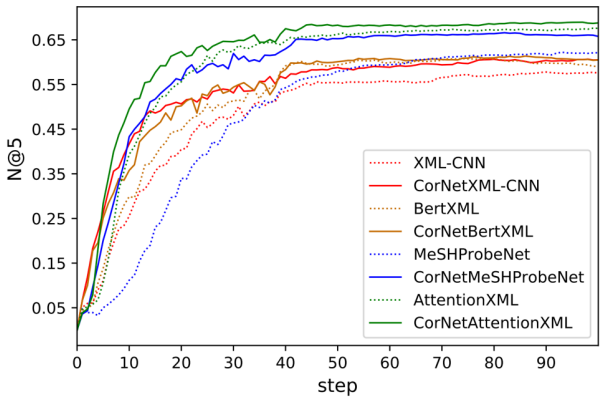


Figure 4: Training curves on EUR-Lex. Dotted lines denote basic models and solid lines denote CorNet models.

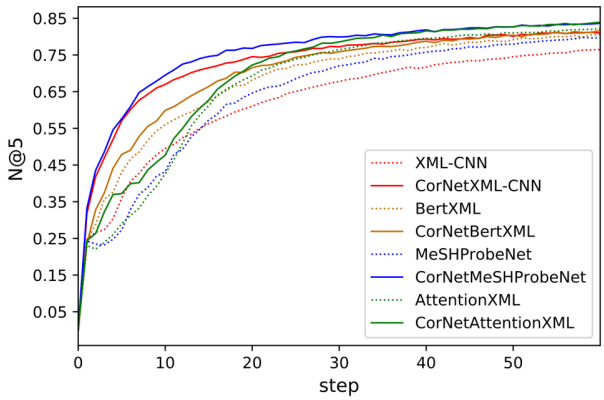


Figure 5: Training curves on AmazonCat-13K. Dotted lines denote basic models and solid lines denote CorNet models.

feed-forward networks in BERT. Although the hidden state of each special symbol only carries a part of the sequence features, this structure enables BertXML to exploit the correlations among special symbols [CLS_1], [CLS_2], ... , [CLS_M] at each layer, resulting in smaller improvements by the CorNet module. Among all the datasets, CorNet favors larger datasets because there exist more label correlations for CorNet to utilize in larger datasets. Among all evaluation metrics, the improvement is more significant for k=5 than for k=1. That is because k=1 represents the most confident label prediction. Rather than benefit from label correlations, the most confident prediction works more often as an anchor to augment the predictions of other labels with label correlations.

AttentionXML is the current state-of-the-art model. As reported in [38], it achieved the highest evaluation scores in comparison with other XMTC models, including many traditional non-deep models, such as AnnexML [30], DiSMEC [2], PfastreXML [15], Parabel [27] and Bonsai [18]. We show that CorNetAttentionXML is able to outperform AttentionXML and achieve the new state-of-the-art results by incorporating label correlations. CorNetBertXML and CorNetMeSHProbeNet also outperformed AttentionXML on AmazonCat-13K.

In addition, as can be seen, the CorNet enhanced models are able to maintain similar training time consumptions and model sizes to their original counterpart models, thanks to the simple architecture of CorNet.

We also observe that from (CorNet)XML-CNN, (CorNet)BertXML to (CorNet)MeSHProbeNet, (CorNet)AttentionXML, the precision grows higher and higher, but the training time consumption also becomes longer and longer. The high efficiency of (CorNet)XML-CNN and (CorNet)BertXML is a result of the fact that the computation of CNNs and BERT is more parallelizable than RNNs. However, the performance scores do not necessarily mean CNNs and BERT are inferior to RNNs in terms of sequence modeling. For example, [4] shows that a carefully designed CNN architecture can achieve comparable performance as RNNs on sequence modeling tasks. But in order to be more consistent with the XMTC literature, we stick to the original XML-CNN architecture instead of switching to the

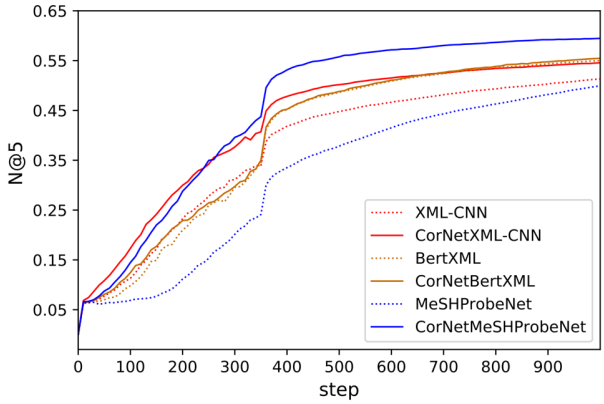


Figure 6: Training curves on Wiki-500K. Dotted lines denote basic models and solid lines denote CorNet models.

CNN architecture in [4]. BERT has also demonstrated its power on a variety of NLP tasks, but in order to make the comparison more fair, we did not use a pretrained large-scale BERT. This limits the strength of BertXML. (CorNet)XML-CNN and (CorNet)BertXML have the potential to attain better performances by adopting more complex designs, expanding model sizes and utilizing pretrained models. Since the focus of this paper is specifically on making use of label correlations, we adhere to the vanilla and non-pretrained models.

5.5 Convergence Study

We have shown that the CorNet models need slightly longer training time than the basic models. That is because each training step takes the CorNet models longer to finish than the basic models. Thus, given a fixed number of training steps, the total training time for the CorNet models would be longer. But in practice, the CorNet architecture is able to help deep XMTC models converge faster. An

Table 5: Performance comparison on model size.

Dataset	model	P@1	P@3	P@5	size (GB)
EUR-Lex	Large	81.35	66.93	56.23	0.32
	CorNet	83.47	70.50	58.73	0.32
AmazonCat-13K	Large	95.67	81.92	66.56	0.96
	CorNet	96.10	82.82	67.57	0.99
Wiki-500K	Large	65.35	44.60	34.87	2.32
	CorNet	69.79	50.68	38.98	2.45

example is CorNetBertXML in Table 3, whose training time is even shorter than BertXML. The reason is that CorNetBertXML triggered the early stopping mechanism and completed training before the designated number of training epochs due to fast convergence.

The training curves of each model on different datasets are depicted in Figures 4, 5 and 6. The x-axis represents the training steps and the y-axis represents the N@5 scores on the validation set. The basic models are denoted by dotted lines and the CorNet models are denoted by solid lines. Lines of the same color are used to represent a model pair, e.g., BertXML and CorNetBertXML. As we can see, CorNet significantly increases the convergence speed of all basic models and improves the final N@5 scores on the validation set.

On the EUR-Lex dataset shown in Figure 4, XML-CNN and MeSHProbeNet converge relatively slow among all the models, while the convergence speed of CorNetXML-CNN and CorNetMeSHProbeNet is greatly escalated. On the AmazonCat-13K dataset shown in Figure 5, all basic models except for BertXML get stuck in the early stage of the training, while the CorNet models are able to train smoothly and rapidly in the beginning. On the Wiki-500K dataset shown in Figure 6, CorNetXML-CNN and CorNetMeSHProbeNet, especially CorNetMeSHProbeNet, converge much faster than their basic counterpart models.

5.6 Ablation Analysis

We also conduct ablation experiments to gain a better understanding of CorNet. The CorNetMeSHProbeNet model is used as the backbone architecture here. The training procedure remains the same as specified in Section 5.1.

5.6.1 Model Size. Since the CorNet module introduces several additional layers into the base module, one might wonder whether the performance improvement is a consequence of the label correlation or simply the additional parameter size. To make the comparison more convincing, we include MeSHProbeNet-large, which is a wider and deeper version of MeSHProbeNet. In order to minimize the influence of model size, MeSHProbeNet-large is configured to have the same size as CorNetMeSHProbeNet. Specifically, the RNN hidden size is increased from 300 to 400, the number of probes is increased from 5 to 10 and the depth of RNN layers is increased from 2 to 3. The comparison result is reported in Table 5, where Large denotes MeSHProbeNet-large and CorNet denotes CorNetMeSHProbeNet. We can see that although MeSHProbeNet-large achieves minor improvement over MeSHProbeNet with the help of a wider and deeper configuration, CorNetMeSHProbeNet is still able to significantly outperform MeSHProbeNet-large.

Table 6: CorNetMeSHProbeNet performance on EUR-Lex with different numbers of CorNet blocks.

#CorNet blocks	P@1	P@3	P@5	N@1	N@3	N@5
1	82.51	70.05	58.40	82.51	73.30	67.57
2	83.47	70.50	58.73	83.47	73.86	67.95
3	83.50	70.35	58.69	83.50	73.67	67.96
4	84.14	70.95	58.98	84.14	74.39	68.38

5.6.2 Number of CorNet Blocks. As mention in Section 3, CorNet is a flexible architecture and an arbitrary number of CorNet blocks can be stacked together to form a deep CorNet module. To investigate the trade-off between performance and computational cost associated with the depth of the CorNet module, we conduct experiments with different numbers of CorNet blocks. As we can see from Table 6, all CorNetMeSHProbeNet models significantly improve over MeSHProbeNet in Table 2, which is equivalent to having 0 CorNet blocks. The performance is robust to the number of CorNet blocks, although more CorNet blocks generally indicate a better performance. 2 CorNet blocks achieve a good balance between performance and complexity and are used as the default setting in the our experiments.

5.7 Case Study

We also conduct a case study to illustrate how CorNet enhances raw label predictions with label correlations. We utilize CorNetMeSHProbeNet and select a document from the test set of the AmazonCat-13K dataset. The product description is "Adidas Vigo Short. ClimaLite short with embroidered logo elastic waist and inner drawcord. 5 inseam. 100% polyester stretch satin. Imported." and it has 7 ground truth labels: "sports & outdoors", "team sports", "soccer", "shorts", "clothing", "men", and "women". The top 10 labels based on their raw predictions are: "sports & outdoors (0.690)", "soccer (0.661)", "team sports (0.302)", "accessories (0.207)", "clothing (0.144)", "lacrosse (0.142)", "athletic (0.139)", "balls (0.133)", "shoes (0.123)", and "home & kitchen (0.099)". The top 10 labels based on their CorNet enhanced predictions are: "sports & outdoors (1.000)", "team sports (1.000)", "soccer (1.000)", "shorts (0.999)", "clothing (0.998)", "men (0.968)", "women (0.620)", "active shorts (0.005)", "balls (0.001)", and "basketball (0.001)". We can see that CorNet is able to promote correlated labels, such as "sports & outdoors", "men", and "women". It is also able to demote uncorrelated labels, such as "accessories" and "lacrosse". Moreover, for the CorNet enhanced predictions, the gap between relevant labels and irrelevant labels, i.e., "women (0.620)" and "active shorts (0.005)", is quite apparent. While for the raw predictions, there is no clear boundary between relevant labels and irrelevant labels. This makes the CorNet enhanced predictions more robust and also sheds some light on the fast convergence speed of CorNet models.

6 CONCLUSION

In this paper we proposed CorNet, an architecture designed to improve deep multi-label text classification models by enabling them to exploit the correlation information among labels. CorNet is a general and independent architecture that can be directly integrated

with any deep XMTC models as an add-on enhancer module. Extensive experimental results demonstrated the effectiveness of CorNet, which is able to advance the state-of-the-art performance on several different multi-label text classification datasets. Moreover, CorNet also exhibited the ability to accelerate the convergence rate during training.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported in part by the US National Science Foundation under grants IIS-1924928, IIS-1938167 and OAC-1934600. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. 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*. 13–24.

[2] Rohit Babbar and Bernhard Schölkopf. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 721–729.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[5] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*. 730–738.

[6] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. 2019. X-BERT: eXtreme Multi-label Text Classification with BERT. *arXiv preprint arXiv:1905.02331* (2019).

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Amit Garg, Jonathan Noyola, Romil Verma, Ashutosh Saxena, and Aditya Jami. 2015. Exploring correlation between labels to improve multi-label classification. *arXiv preprint arXiv:1511.07953* (2015).

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (2018).

[15] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 935–944.

[16] Kalina Jasinska, Krzysztof Dembczynski, Róbert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hullermeier. 2016. Extreme F-measure maximization using sparse probability estimates. In *International Conference on Machine Learning*. 1435–1444.

[17] Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. 2018. AttentionMeSH: Simple, Effective and Interpretable Automatic MeSH Indexer. In *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*. 47–56.

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

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).

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

[22] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. 165–172.

[23] Eneldo Loza Mencia and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 50–65.

[24] Tien Thanh Nguyen, Thi Thu Thuy Nguyen, Anh Vu Luong, Quoc Viet Hung Nguyen, Alan Wee-Chung Liew, and Bela Stantic. 2019. Multi-label classification via label correlation and first order feature dependance in a data stream. *Pattern recognition* 90 (2019), 35–51.

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

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[27] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*. 993–1002.

[28] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 263–272.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[30] Yukihiro Tagami. 2017. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 455–464.

[31] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artieres, Axel Ngonga, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics* 16 (2015), 138. https://doi.org/10.1186/s12859-015-0564-6

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[33] Guangxu Xun, Kishlay Jha, Ye Yuan, Yaqing Wang, and Aidong Zhang. 2016. MeSHProbeNet: A Self-attentive Probe Net for MeSH Indexing. *Bioinformatics* 32, 12 (2016), 70–79. https://doi.org/10.1093/bioinformatics/btw294

[34] Guangxu Xun, Yaliang Li, Jing Gao, and Aidong Zhang. 2017. Collaboratively improving topic discovery and word embeddings by coordinating global and local contexts. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 535–543.

[35] Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. 2017. A Correlated Topic Model Using Word Embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

[36] Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. *arXiv preprint arXiv:1806.04822* (2018).

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

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