IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 26, 2019, accepted December 12, 2019, date of publication December 18, 2019,
date of current version December 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960626

Incorporating Label Co-Occurrence Into Neural Network-Based Models for Multi-Label Text Classification

JIAQI YAO, KEREN WANG, AND JIKUN YAN
National Key Laboratory of Science and Technology on Blind Signal Processing, Chengdu 610041, China
Corresponding author: Jiaqi Yao (jiaqi_yao@163.com)

ABSTRACT Multi-label text classification (MLTC) addresses a fundamental problem in natural language processing, which assigns multiple relevant labels to each document. In recent years, Neural Network-based models (NN models) for MLTC have attracted much attention. In addition, NN models achieve favorable performances because they can exploit label correlations in the penultimate layer. To further capture and explore label correlations, we propose a novel initialization to incorporate label co-occurrence into NN models. First, we represent each class as a column vector of the weight matrix in the penultimate layer, which we name the class embedding matrix. Second, we deduce an equation for correlating the class embedding matrix with the label co-occurrence matrix, ensuring that relevant classes are denoted by vectors with large correlations. Finally, we provide a theoretical analysis of the equation, and propose an algorithm to calculate the initial values of the class embedding matrix from the label co-occurrence matrix. We evaluate our approach with various text extractors, such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN) and Transformer on four public datasets. The experimental results demonstrate that our approach markedly improves the performance of existing NN models.

INDEX TERMS Multi-label text classification, label co-occurrence, initialization, neural network, class embedding.

I. INTRODUCTION

In many practical applications, a document is often labeled with multiple labels. For example, descriptive texts about books, music or videos may be labeled with more than one label, and a news document may belong to several topics [1], [2]. Multi-label text classification (MLTC) addresses the problem by learning a mapping function from a document to a relevant subset of the whole classes.

Label correlations are informative to MLTC [3], [4]. For example, if a movie's descriptive text is labeled with ''Kung Fu'', it is usually also labeled with ''Chinese''. Many models exploit label correlations to improve MLTC performance [3]–[8]. These models can be roughly divided into three categories based on the order of label correlation they exploit: (1) first-order, which ignores the correlation of labels; (2) second-order, which considers the pairwise correlation between labels; (3) high-order, which impose influence of all other labels on each label [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Yanzheng Zhu.

In recent years, considerable progress has been made in deep learning, and NN models have had a dramatic impact on many fields of machine learning [14]. There are also many NN models that address the problem of MLTC. The key difference among different NN models for MLTC is the text feature extractor [49], such as Convolutional Neural Network (CNN, [22]), Recurrent Neural Network (RNN, [28], [47], [48]) and Transformer [33]. In addition, these NN models achieve favorable performances because they can exploit label correlations in the penultimate layer by training [2].

To further explore label correlations for NN models, we propose a novel initialization to incorporate label co-occurrence into NN models. The penultimate layer of NN models is close to the output, which suggests there is more class-specific information in the layer and it is easier to learn class-specific weights in the layer. Thus, just as word embedding represents a word as a continuous dense vector, we represent each class as a column vector of the weight matrix in the penultimate layer of NN models, which we name the class embedding matrix. And the class probability of an example is calculated by the correlation between its

feature vector and the class vector. The motivation for our approach is based on the fact that the more examples the two classes are both associated with, the closer the two corresponding class vectors are. With this motivation, we derive an equation between the class embedding matrix and the label co-occurrence matrix. The equation correlates the included angle between two class vectors with the co-occurrence times of the two classes. After theoretical analysis, we find that the equation does not necessarily have an analytical solution. Therefore, we propose an approximate algorithm to calculate the initial values of the class embedding matrix from the label co-occurrence matrix with this equation. Finally, we use the initial values to initialize the class embedding matrix. Our approach belongs to the second-order, which uses the pairwise correlation between labels. In addition, we evaluate our approach with CNN, RNN and Transformer on four public datasets. For each text feature extractor we use the state-of-the-art architecture, i.e. XML-CNN for CNN [22], Hierarchal Attention Network for RNN [28] and BERT for Transformer [33]. The experimental results demonstrate that our approach works for the various text feature extractors, and BERT with our approach achieves the best performance.

In summary, the key contributions of this work are as follows:

- We represent each class as a column vector of the weight matrix in the penultimate layer, which we name the class embedding matrix. And then we derive an equation to correlate the label co-occurrence matrix and the class embedding matrix.
- We analyze the properties of the equation between the label co-occurrence matrix and the class embedding matrix, and design an algorithm to calculate the initialized values for the class embedding matrix from the label embedding matrix with the equation.
- We validate the effectiveness of our approach with different text feature extractors on four public datasets. In addition, for each text feature extractor, we adopt state-of-the-art architecture, i.e. TextCNN for CNN, Hierarchical Attention Network for RNN and BERT for Transformer.

II. RELATED WORKS

A. TEXT FEATURE EXTRACTOR

Text feature extractors can be categorized into two families. One represents a word as a one-hot vector, while the other represents a word as a continuous dense vector of fixed length (i.e. word embedding). The text feature extractor based on one-hot, such as tf-idf, cannot model the similarities between words, resulting in a lack of semantic information. However, word embedding, which is trained on a large corpus, can capture semantic information via the co-occurrence of words. And word embedding is a kind of distributed representation [16]. CBOW and Skip-Gram [17]–[20] are two popular methods for training word embedding. For the above advantage, many text classification models based on word embedding have been proposed [15]. These models usually represent text in terms of word embedding and then input into different neural networks. These models can be roughly divided into two categories according to the architecture of the neural network they use: models based on CNN [21]–[24] and RNN [28], [29].

Recently, some researchers have argued that a word has different meanings in different contexts [30], and have thus proposed contextualized word-embedding and various fine-tuning approaches, such as ULMFiT [31], GPT [32], and BERT [33]. These models pre-train language models on certain language model objectives (e.g. the next word prediction), and then fine-tune a supervised downstream task. Among these models, the basic block of GPT and BERT is Transformer, which is composed of attention mechanism [25]. These achieve state-of-the-art results in many NLP tasks.

B. MULTI-LABEL CLASSIFICATION MODELS

Multi-label classification is to learn a function to associate an example with multiple relevant labels. In addition, these models can be divided into two families: i.e., problem transformation models and algorithm adaptation models [1], [34]:

Problem transformation models fit data to algorithms. Models representative of this type include Binary Relevance [35], Classifier Chains [36], Calibrated Label Ranking [37] and Label Power [38]. Binary Relevance transforms multi-label classification into many binary classifiers, and thus does not exploit any correlation among classes. Classifier Chain constructs a chain of binary classifiers that receives the output of previous classifiers as partition input. Calibrated Label Ranking constructs a set of classifiers between pairing classes. Label Power directly transforms multi-label into multi-class classification.

Algorithm transformation models fit algorithms to data. Models representative of this type ML-kNN [39], ML-DT [40], Rank-SVM [41], Modified NN models [2], [11]. ML-kNN adapts kNN to multi-label classification in the framework of Bayesian probabilities. ML-DT primarily modifies the criterion of node splitting in the decision tree model to adapt to the multi-label classification. Rank-SVM modifies the SVM objective function by maximizing the distance between relevant and irrelevant labels. NN models also modify the loss function (e.g. the pairwise loss function, the binary cross entropy function) for adaptation.

III. APPROACH

In this section, we present our approach, which aims to incorporate label co-occurrence into NN models. First, we briefly review the neural network-based models for MLTC. Then, we present the design of the label co-occurrence matrix, and derive the corresponding equation with the weight matrix in the penultimate layer, which we name the class embedding matrix. Finally, we provide some theoretical analysis of the matrix decomposition that the equation involves and propose an algorithm to calculate the initial values of the class embedding matrix from the label co-occurrence matrix. For ease

TABLE 1. Notations.

Notations	Mathematical Meanings
$\|X\|$	The cardinality of set X
R^N	N-dimension real number field
$\{0,1\}^D$	D-dimension vector composed of 0 and 1
$\sigma(z)$	Sigmoid function $\frac{1}{1+exp(-z)}$
A^T	Matrix Transpose of A
$\|v\|_2$	2-norm of vector v
$det(A)$	The determinant of Matrix A
$\|A\|_F$	Frobenius norm of A
$I[\cdot]$	The indicator function
$diag(a_1, a_2, \ldots, a_n)$	A square, diagonal matrix with diagonal entries given by $a_1, a_2, \ldots, a_n$

of reference, we summarize major notations throughout this paper in Table 1. In addition, all matrices are valid for real numbers.

A. NEURAL NETWORK-BASED MODELS FOR MULTI-LABEL TEXT CLASSIFICATION

We denote an example set by X and let $L = \{l_1, l_2, \ldots, l_{|L|}\}$ be the class set. In MLTC, an example $x_i \in X (1 \leq i \leq |X|)$ is labeled with multiple classes in L, which we denote by $y_i \in Y \subseteq \{0,1\}^{|L|}$, where $y_{ij} = 1 (1 \leq j \leq |L|)$ if x_i is with the class l_j. The task of MLTC is to learn from the training dataset $\{X, Y\}_{training}$ to approximate a mapping function $f : x_i \rightarrow y_i$.

Rank loss is one of the MLTC loss functions that describes the average fraction of reverse-ordering label pairs between the relevant and irrelevant labels. Given an example $(x, y) \in (X, Y)$, for the mapping function f the rank loss is defined as follows [9]:

$$L_{rnk}(f, (x, y)) = c(y) \sum_{y_p < y_q} I\left[f(x)_p > f(x)_q\right] + \frac{1}{2} I\left[f(x)_p = f(x)_q\right] \quad (1)$$

where $c(y)$ is the cost for a mistake which may depend on properties of ground truth y, and $f(x)_p$ is the prediction score for label p.

Unfortunately, the rank loss is nonconvex and is thus, difficult to minimize. However, two kinds of convex surrogate loss functions for NN models are proposed, and proven to be consistent with the rank loss [9], [10]. One is the following pairwise surrogate loss function [9]:

$$J_{pw}(f, (x, y)) = \frac{1}{|y_r|\,|y_{ir}|} \sum_{(p,q) \in y_r \times y_{ir}} \exp\left(-\left(f(x)_p - f(x)_q\right)\right) \quad (2)$$

where y_r is the relevant subset of y, and y_{ir} is the irrelevant subset of y. The pairwise rank loss function is implemented in the algorithm called BP-MLL [11].

The other is the following univariate surrogate loss function [10]:

$$J_{uni}(f, (x, y)) = c(y) \sum_{p=1}^{|L|} \log\left(1 + \exp\left(-y_p^* f(x)_p\right)\right) \quad (3)$$

where $y_p^* \in \{-1, 1\}$. If we set $c(y) = 1$ and apply sigmoid activation function in the output layer, then we get the Binary Cross Entropy (BCE) loss function [2]:

$$J_{BCE}(f, (x, y)) = -\sum_{p=1}^{|L|} \left(y_p \log\left(\sigma\left(f(x)_p\right)\right) + \left(1 - y_p\right) \log\left(1 - \sigma\left(f(x)_p\right)\right)\right. \quad (4)$$

The BCE loss function has been proven to be consistent with the rank loss [2], [10]. Additionally, many NN models use the BCE loss function because of its simplicity and effectiveness [12]. Therefore, our approach is based on the BCE loss function.

B. EQUATION BETWEEN LABEL CO-OCCURRENCE MATRIX AND CLASS EMBEDDING MATRIX

Our approach takes the original NN models as the basic models. We first extract a feature vector $z : D \times 1$ from a text via a feature extractor. The feature extractor can be CNN, RNN or Transformer. Then, just as the word embedding matrix represents each word as a continuous vector, we represent each class as each column vector of weight matrix $W : D \times |L|$ in the penultimate layer. Let $o : |L| \times 1$ equal to $W^T z$. Finally, we set $p = sigmoid(o)$, where p is with the dimension of $|L| \times 1$. Additionally, *sigmoid* is an element-wise function, i.e. $p_i = \frac{1}{1+exp(-o_i)}, 1 \leq j \leq |L|$. NN models with BCE loss function for MLTC are illustrated in Fig. 1.

Typically, if $p_i > 0.5$, the input text is determined to be associated with the corresponding class l_i. And if $p_i > 0.5$, then $o_i > 0$, which means ${w_i}^T \cdot z > 0$, where w_i is the i-th column of the class embedding matrix W. Thus, on $\{X, Y\}_{training}$, we get the following:

$$A_{ij} = P\left({w_i}^T \cdot z > 0 \text{ and } {w_j}^T \cdot z > 0\right) = \frac{\#(i,j)}{N} \quad (5)$$

$$B_{ij} = P\left({w_i}^T \cdot z > 0 \text{ or } {w_j}^T \cdot z > 0\right) = \frac{\#(i) + \#(j) - \#(i,j)}{N} \quad (6)$$

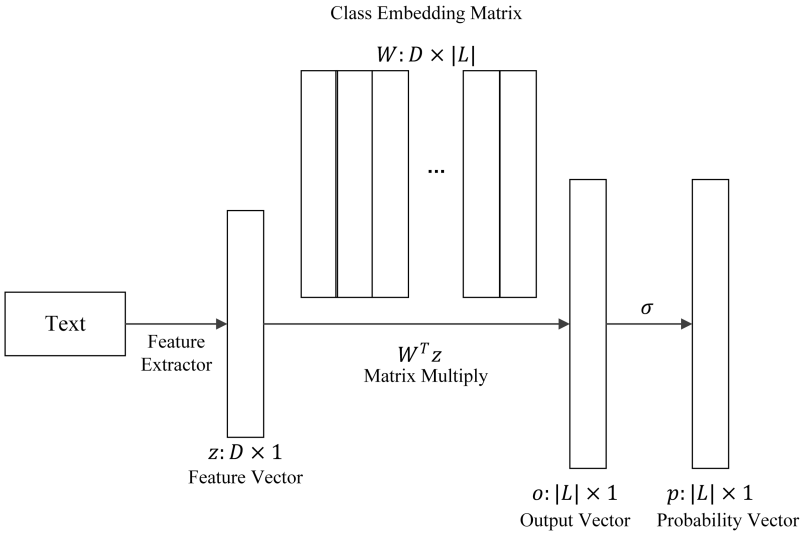

FIGURE 1. NN models for MLTC.

$$C_{ij} = \frac{A_{ij}}{B_{ij}} = \frac{P\left(w_i \cdot z > 0 \text{ and } w_j \cdot z > 0\right)}{P\left(w_i \cdot z > 0 \text{ or } w_j \cdot z > 0\right)} = \frac{\#(i,j)}{\#(i) + \#(j) - \#(i,j)} \tag{7}$$

where N is the number of training examples, $\#(i,j)$ is the number of examples that are associated with both l_i and l_j, and $\#(i)$ is the number of examples that are with l_i. Compared with (5) and (6), (7) has an advantage where the numerator and denominator have the same order of magnitude, guaranteeing the stability of the numerical calculation.

Let θ_{ij} denote the included angle between w_i and w_j. ${w_i}^{\perp}$ and ${w_j}^{\perp}$ are respectively the vertical vector of w_i and w_j. We refer to the included angle between w_i and z as θ, and based on the property of cosine function, we get the following:

$$w_i \cdot z = \|w_i\|_2 \|z\|_2 \cos(\theta) > 0, \quad 0 < \theta < \frac{\pi}{2} \tag{8}$$

As shown in Fig. 2, and according to (8), we get the following:

$$P\left(w_i \cdot z > 0 \text{ and } w_j \cdot z > 0\right) = \frac{(\pi - \theta_{ij})}{(2\pi)} \tag{9}$$

$$P\left(w_i \cdot z > 0 \text{ or } w_i \cdot z > 0\right) = \frac{(\pi + \theta_{ij})}{(2\pi)} \tag{10}$$

$$\frac{P\left(w_i \cdot z > 0 \text{ and } w_j \cdot z > 0\right)}{P\left(w_i \cdot z > 0 \text{ or } w_j \cdot z > 0\right)} = \frac{(\pi - \theta_{ij})}{(\pi + \theta_{ij})} \tag{11}$$

Substituting (7) into (11), we get the following:

$$\theta_{ij} = \frac{\pi(1 - C_{ij})}{1 + C_{ij}} \tag{12}$$

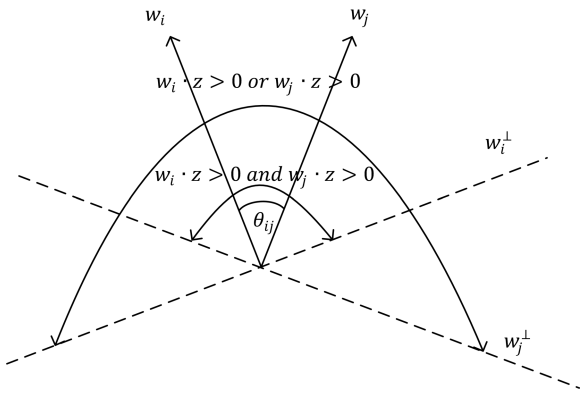

FIGURE 2. Geometric Illustration of (9)-(11).

Let $Z_{ij} = \cos(\theta_{ij})$, then we get the following:

$$Z_{ij} = \cos\left(\frac{\pi(1 - C_{ij})}{1 + C_{ij}}\right) = \cos\left(\theta_{ij}\right) = \frac{w_i \cdot w_j}{\left(\|w_i\|_2 \left\|w_j\right\|_2\right)} \tag{13}$$

Expressing (13) in the form of a matrix yields the following:

$$Z = \cos(\frac{\pi(1 - C)}{(1 + C)}) = W^T W \tag{14}$$

where Z is composed of $Z_{ij} (1 \le i, j \le |L|)$, C is composed of $C_{ij} (1 \le i, j \le |L|)$. For that $\|w_i\|_2 = {w_i}^T w_i = c_{ii} = 1$, (14) is without normalization.

Equation (14) is the key result of our approach, which correlates the class embedding matrix W with the label co-occurrence matrix X. The equation ensures that the more examples the two classes share, the closer the distance of their corresponding vectors, and vice versa. In addition, the label

Input: Z
Output: W
Let $\delta = 1e-9, error_{pre} = +\infty, \epsilon = 1e-3$
$error = \|W^T W - Z\|_F$
while $|error - error_{pre}| > \delta$:
  $error_{pre} = error$
  $W = W - \epsilon \frac{\partial error}{\partial W}$
  $error = \|W^T W - Z\|_F$

FIGURE 3. Algorithm to approximate the decomposition of Z into $W^T W$.

co-occurrence matrix Z can be counted and calculated from the training dataset. Thus, the initial values of the class embedding matrix can be calculated from Z, which we denote by W^*, and the theoretical analysis on the calculation is deferred to III-C. Finally, we use W^* to initialize W.

C. ANALYSIS ON THE DECOMPOSITION OF THE LABEL CO-OCCURRENCE MATRIX Z

Equation (14) involves the decomposition of the label Co-occurrence matrix Z into $W^T W$. In this section, we analyze whether the properties of matrix Z ensure that the decomposition exists.

According to the definition of Z, the following properties hold:

(1) Z is a symmetric real matrix.
(2) $|Z_{ij}| \leq 1$
(3) $Z_{ii} = 1$

The decomposition of Z into $W^T W$ exists, if and only if Z is positive definite or positive semi-definite [13]. However, the properties of Z cannot ensure that Z is positive definite or positive semi-definite. Let us take a 3×3 matrix M_3 for example:

$$\begin{pmatrix} 1 & a & b \\ a & 1 & c \\ b & c & 1 \end{pmatrix}$$

where $-1 \leq a, b, c \leq 1$. $det(M_3) = 1 - a^2 - b^2 - c^2 + 2abc$. If M_3 is positive definite or positive semi-definite, then $det(M_3) \geq 0$ is necessary [13]. However, the value of $det(M_3)$ depends on the value of a, b, c, and it may be less than 0, for example, when $a = b = c = -1$, $det(M_3) = -4 < 0$.

Therefore, the decomposition of Z into $W^T W$ does not necessarily exist. So we propose an algorithm to approximate the decomposition, which is shown in Fig. 3. We design an objective function $error = \|W^T W - Z\|_F$, which is minimized by the gradient descent algorithm. If the decomposition exists, then $error = 0$. If the decomposition does not exist, the algorithm shown in Fig.3 makes $W^T W$ approximate Z by minimizing $error$, and ensures the properties of Z as much as possible. When minimizing $error$, take w_i and w_j for example, $w_i \cdot w_j$ approximates z_{ij}. Thus W satisfies that the more examples the two classes are both associated with, the closer the two corresponding class vectors are.

TABLE 2. A summary of dataset.

datasets	$\#train$	$\#valid$	$\#test$	$\|L\|$	$\overline{L}$
Reuters-21578	7,002	778	3,022	118	1.24
RCV1-v2	20,835	2,314	781,265	103	3.24
AG-multi	74,784	9,348	9,348	12	2.74
20newsgroup	10,183	1,131	7,532	33	2.75

IV. EXPERIMENTS

In this section, the four public datasets used in this study are first introduced. Then, details of how to implement our approach with different text feature extractors are presented. Finally, experimental results are reported and discussed.

A. DATASET

We collect four public MLTC datasets to evaluate our approach.

Reuters-21578. Reuters-21578 consists of 10802 texts, which is often used for MLTC evaluation [42].

RCV1-v2. RCV1-v2 is made up of many documents, including 20,835 training documents and 781,265 test documents [43] within 103 classes.

AG-multi. We obtain the AG corpus on the web,[1] which contains 496,835 classified news articles. We collect news documents that are classified by multiple labels to construct the multi-label text datasets. In addition, any class with less than 100 documents is removed. Finally, we get the AG-multi dataset of 93,480 news articles within 12 classes. We randomly select 80% of AG-multi for training, 10% for validating and 10% for testing.

20newsgroup. The 20newsgroup[2] dataset is roughly evenly partitioned into 20 groups, such as 'talk.politics.misc', 'rec.sport.hockey', etc. We split the newsgroups by '.', for example, 'talk.politics.misc' is split into 'talk', 'politics' and 'misc'. And then, we get a collection of multi-label documents.

For Reuters-21578, RCV1-v2, and 20newsgroup, we reserve 10% of the training dataset for validation. The statistics of these datasets are summarized in Table 2, where *#train*, *#valid* and *#test* represent the corresponding number of examples, $|L|$ represents the number of total classes, and $\overline{L}$ represents the average of labels per example.

B. IMPLEMENTATION DETAILS

In this work, we take CNN, RNN, and Transformer as the text feature extractors. Next, we elaborate on three state-of-the-art models that use each of these text feature extractors. For each model, we use BCE as the loss function. And for simplicity, we do not introduce the output layer, which has already been introduced and described before.

1) CONVOLUTIONAL NEURAL NETWORK

For CNN, we adopt XML-CNN as the basic architecture [12]. The whole architecture is shown in Fig. 4. We first

[1] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, accessed on July,10,2019
[2] http://qwone.com/ jason/20Newsgroups/, accessed on July,10,2019

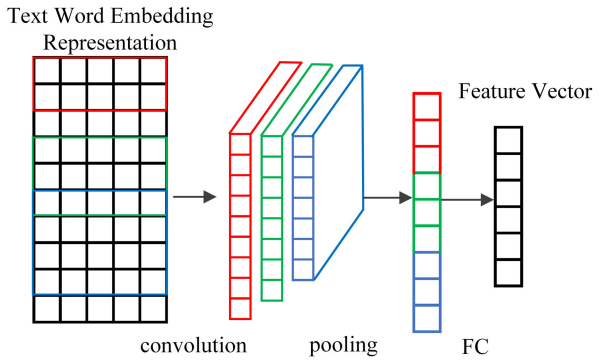

FIGURE 4. XML-CNN.

represent the text with the pre-trained word vectors, and then apply convolutional and pooling operations. We then add a fully-connected (FC) layer for controlling the dimension of the feature vector v.

For all datasets, we use the following: pre-trained word2vec vectors that are from [19] (words that do not appear in pre-trained word2vec vectors are initialized randomly following a uniform distribution); convolution kernel size of $3 \times d$, $4 \times d$ and $5 \times d$, where d is the dimension of pre-trained word2vec vectors; 256 convolution kernels for each size, rectified linear activation function; dropout rate of 0.5 for regularization. The dropout layer is added before the convolutional layer. In addition, the dimension of v equals the number of total class for each dataset. We adopt Adam [45] to optimize the parameters, and the learning rate is determined by the performance on the validating dataset.

2) RECURRENT NEURAL NETWORK

We use Hierarchical Attention Networks (HAN, [28]) as the basic model for RNN. HAN applies both word-level and sentence-level attention mechanism. In this study, we only apply the word-level attention mechanism, and the whole architecture is shown in Fig. 5. We first embed the words to vectors $v_1, v_2, \cdots, v_T$ through the pre-trained word embedding matrix. Then, we use a bidirectional GRU and word attention to get the representation of a text. GRU stands for Gated Recurrent Unit, which is a kind of variant of RNN. The whole calculation is listed specifically as follows:

$$\begin{aligned}
\overrightarrow{h_t} &= \overrightarrow{GRU}\,(v_t)\,, \quad t \in [1, T] \\
\overleftarrow{h_t} &= \overleftarrow{GRU}\,(v_t)\,, \quad t \in [1, T] \\
h_t &= \left[\overrightarrow{h_t}\,, \overleftarrow{h_t}\right] \\
u_t &= \tanh\left(Wh_t + b\right) \\
\alpha_t &= \frac{\exp\left(u_t^T u_w\right)}{\sum \exp\left(u_t^T u_w\right)} \\
s &= \sum_t \alpha_t h_t
\end{aligned}$$

Finally, we input s into an FC layer to get the feature vector v. In addition, the detailed settings of RNN for MLTC are the same pre-trained word2vec vectors and dimension of the feature vector similar to those in CNN, 128 units in the GRU cell, hyperbolic tangent activation function, and the dropout rate of 0.5. The dropout layer is added before the GRU layer, and the optimization is the same as that used in CNN.

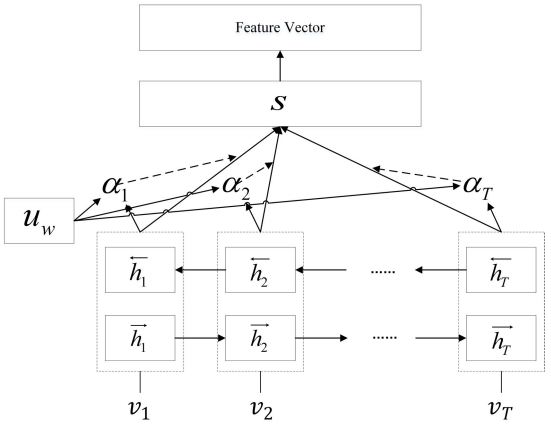

FIGURE 5. Hierarchical attention network.

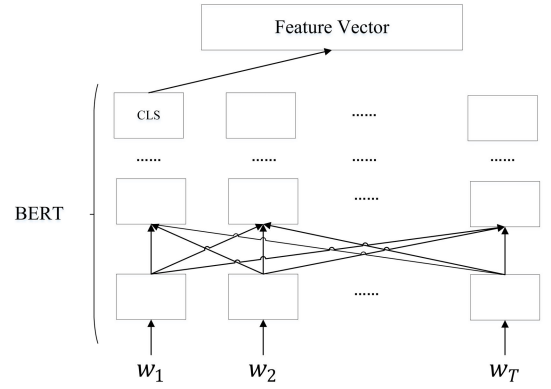

FIGURE 6. BERT.

3) TRANSFORMER

For Transformer, we adopt BERT, which is a language representation model [33] and achieves many state-of-the-art results for many NLP tasks. In this study, we input the special classification embedding ([CLS]) of BERT into an FC layer to get the feature vector v. The whole architecture is shown in Fig. 6.

We use the pre-trained BERT model uncased_L-12_H-768_A-12,[3] which has 12 layers Transformer blocks, 768 hidden units, 12 multi-heads, and 110M parameters in total. Uncased means that the text has been lowercased. We also set the dimension of v to be the number of total classes for each dataset. We adopt the same optimization in the original BERT paper.

C. EXPERIMENTAL RESULTS

We report the detailed experimental results on four datasets in Table 3, Table 4, Table 5 and Table 6. The definition of

[3]https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip, accessed on July,10,2019

TABLE 3. Results on reuters.

Metrics	XML-CNN	XML-CNN*	HAN	HAN*	BERT	BERT*
Micro-Precision	0.8694	0.8773	0.8879	0.8856	**0.9259**	0.9176
Micro-Recall	0.8778	0.8869	0.8513	0.8816	0.8706	**0.8987**
Micro-F1	0.8736	0.8821	0.8692	0.8836	0.8974	**0.9081**
Marco-Precision	0.3940	0.4440	0.4137	0.4456	0.4387	**0.5380**
Marco-Recall	0.3488	0.3899	0.3493	0.3592	0.3377	**0.4534**
Marco-F1	0.3700	0.4152	0.3788	0.3977	0.3816	**0.4921**
Hamming Loss	0.0028	0.0026	0.0028	0.0025	0.0022	**0.0020**
Accuracy	0.8847	0.8928	0.8835	0.8969	0.9125	**0.9231**
Precision	0.9109	0.9169	0.9140	0.9234	0.9313	**0.9490**
Recall	0.9216	0.9289	0.9063	0.9298	0.9461	**0.9484**
F1	0.9055	0.9130	0.9014	0.9168	0.9296	**0.9403**

TABLE 4. Results on RCV1-v2.

Metrics	XML-CNN	XML-CNN*	HAN	HAN*	BERT	BERT*
Micro-Precision	0.8365	0.8238	0.8077	0.7996	**0.8662**	0.8448
Micro-Recall	0.7449	0.7668	0.6758	0.6960	0.7144	**0.7849**
Micro-F1	0.7880	0.7942	0.7359	0.7443	0.7830	**0.8137**
Marco-Precision	0.6431	0.6485	0.4336	0.4463	0.4808	**0.6687**
Marco-Recall	0.4292	0.4737	0.3028	0.3178	0.2948	**0.4924**
Marco-F1	0.5148	0.5475	0.3565	0.3712	0.3655	**0.5672**
Hamming Loss	0.0129	0.0128	0.0156	0.0154	0.0127	**0.0115**
Accuracy	0.7205	0.7312	0.6672	0.6723	0.7220	**0.7696**
Precision	0.8656	0.8541	0.8327	0.8260	0.7609	**0.8289**
Recall	0.7875	0.8091	0.7240	0.7393	**0.8842**	0.8682
F1	0.7959	0.8052	0.7480	0.7540	0.7930	**0.8310**

TABLE 5. Results on AG-multi.

Metrics	XML-CNN	XML-CNN*	HAN	HAN*	BERT	BERT*
Micro-Precision	0.7657	0.7828	0.7945	0.7820	**0.7976**	0.7886
Micro-Recall	0.8801	0.8625	0.8099	0.8772	0.8468	**0.8852**
Micro-F1	0.8189	0.8207	0.7945	0.8269	0.8215	**0.8341**
Marco-Precision	0.8051	0.8154	**0.8292**	0.8161	0.7683	0.7619
Marco-Recall	0.8430	0.8625	0.6962	0.8485	0.8527	**0.8858**
Marco-F1	0.8236	**0.8383**	0.7569	0.8320	0.8083	0.8192
Hamming Loss	0.0686	0.0664	0.0738	0.0647	0.0649	**0.0621**
Accuracy	0.7320	0.7342	0.6918	0.7413	0.7411	**0.7616**
Precision	0.7786	0.7943	0.7921	0.7926	0.8568	**0.8940**
Recall	**0.8737**	0.8621	0.8664	0.8718	0.8188	0.8160
F1	0.7956	0.7975	0.7993	0.8022	0.8083	**0.8256**

TABLE 6. Results on 20newsgroup.

Metrics	XML-CNN	XML-CNN*	HAN	HAN*	BERT	BERT*
Micro-Precision	0.8155	0.8510	0.7843	0.7820	**0.8898**	0.8875
Micro-Recall	0.7484	0.7674	0.8740	0.8772	0.8736	**0.8949**
Micro-F1	0.7805	0.8071	0.8267	0.8269	0.8816	**0.8912**
Marco-Precision	0.8126	0.8416	0.8273	0.8161	0.8770	**0.8756**
Marco-Recall	0.7016	0.7336	0.8488	0.8485	0.8548	**0.8753**
Marco-F1	0.7530	0.7839	0.8379	0.8320	0.8658	**0.8755**
Hamming Loss	0.0350	0.0305	0.0646	0.0647	0.0195	**0.0182**
Accuracy	0.7019	0.7405	0.7393	0.7413	0.8633	**0.8717**
Precision	0.8130	0.8448	0.7921	0.7926	0.8777	**0.8950**
Recall	0.7544	0.7789	0.8664	0.8718	0.8860	**0.8918**
F1	0.7586	0.7873	0.7993	0.8022	0.8791	**0.8903**

evaluation metrics can be found in [44]. In addition, the best result is in boldface. The algorithms marked with * use the proposed initialization in this work, and others use the same initialization in the corresponding original paper.

To compare the performance of different approaches across multiple datasets, we apply the corrected Friedman test and the post-hoc Nemenyi test as recommended by Demšar [46]. Friedman test is used to determine whether the performances of different approaches are equal. If a statistically significant difference is detected, Nemenyi test is then used to further distinguish the approaches. For brevity, we only conduct Friedman test on Micro-F1, Marco-F1, Hamming

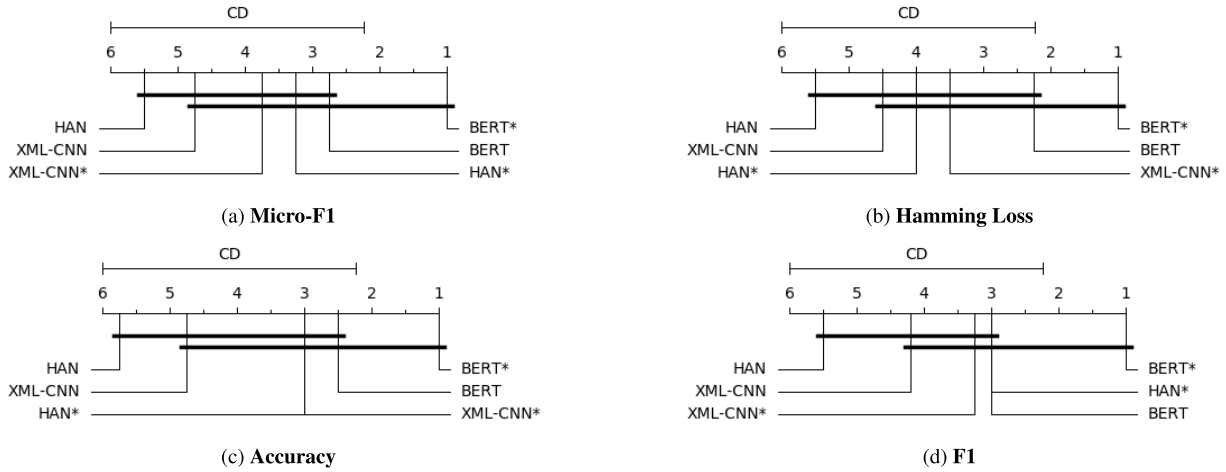

FIGURE 7. CD diagrams.

Loss, Accuracy and F1. Note that Micro-F1, Marco-F1 and F1 are composite evaluation metrics of respective precision and recall. For these evaluation metrics except Marco-F1, Friedman test at 0.05 significance level rejects the hypothesis of equal performance. Rare classes have a great influence on Marco-F1, and all these approaches perform approximately equally on the rare classes. Therefore, we further conduct Nemenyi test on Micro-F1, Hamming Loss, Accuracy and F1. Specifically, Nemenyi test figures out the Critical Difference (CD) based on the significance level for average ranks and the number of datasets. The results of the Nemenyi test are presented in Fig. 7. The average ranks are shown along the horizontal axis, and $CD = 3.770$ is also shown above the axis.

To summarize, we draw the following conclusions:

- The proposed initialization improves the performance of models with different text feature extractors on most evaluation metrics, even for the state-of-the-art model BERT. The proposed approach incorporates the label co-occurrence into the neural network models, and uses the label co-occurrence matrix to calculate the initial value of the class embedding matrix, thereby improving the performance of different neural network models on the multi-label text classification.
- For all datasets, BERT yields markedly better results than those of XML-CNN and HAN. The reason is perhaps that BERT has more parameters. But it's more likely that the pre-trained language model in BERT works better on describing characteristics of natural language data than word embedding that CNN and RNN adopt.

V. CONCLUSION

In this study, we propose a novel initialization to incorporate label co-occurrence into neural network-based models for multi-label text classification. We represent each class as a column vector of the weight matrix in the penultimate layer, which we name the class embedding matrix. Because related classes usually share the same instances, which is called label co-occurrence in this study, we derive an equation between the class embedding matrix and the label co-occurrence matrix. We also provide a theoretical analysis of this equation, and propose an algorithm to calculate the initial values of the class embedding matrix from the label co-occurrence matrix with this equation. We evaluate our approach with prevalent text feature extractors, including CNN, RNN and Transformer on four public datasets. In addition, we adopt the state-of-the-art architecture for each text feature extractor, i.e. XML-CNN for CNN, HAN for RNN and BERT for Transformer. The experimental results demonstrate the effectiveness of our approach compared with the original initialization in those models. In the future, we will generalize our approach to other multi-label classification tasks.

REFERENCES

[1] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.

[2] J. Nam, J. Kim, E. L. Mencía, I. Gurevych, and J. Fürnkranz, "Large-scale multi-label text classification—Revisiting neural networks," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2014, pp. 437–452.

[3] Z. F. He and M. Yang, "Sparse and low-rank representation for multi-label classification," *Appl. Intell.*, vol. 49, no. 5, pp. 1708–1723, 2018.

[4] Y. Zhu, J. T. Kwok, and Z.-H. Zhou, "Multi-label learning with global and local label correlation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1081–1094, Jun. 2018.

[5] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, p. 333, Dec. 2011.

[6] S. J. Huang and Z. H. Zhou, "Multi-label learning by exploiting label correlations locally," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012.

[7] W. Bi and J. T. Kwok, "Multilabel classification with label correlations and missing labels," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014.

[8] M. L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 999–1008.

[9] W. Gao and Z. H. Zhou, "On the consistency of multi-label learning," in *Proc. 24th Annu. Conf. Learn. Theory*, 2011, pp. 341–358.

[10] K. Dembczynski, W. Kotlowski, and E. Hüllermeier, "Consistent multilabel ranking through univariate losses," 2012, *arXiv:1206.6401*. [Online]. Available: https://arxiv.org/abs/1206.6401

[11] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1338–1351, Oct. 2006.

[12] J. Liu, W. C. Chang, and Y. Wu, "Deep learning for extreme multi-label text classification," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 115–124.
[13] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[15] Y. Goldberg, "Neural network methods for natural language processing," *Synth. Lect. Hum. Lang. Technol.*, vol. 10, no. 1, pp. 1–309, 2017.
[16] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.
[17] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proc. HLT-NAACL*, vol. 13, 2013, pp. 746–751.
[18] T. Mikolov, K. Chen, and G. Corrado, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781
[19] T. Mikolov, I. Sutskever, and K. Chen, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[20] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, *arXiv:1607.01759*. [Online]. Available: https://arxiv.org/abs/1607.01759
[21] S. Wang, M. Huang, and Z. Deng, "Densely connected CNN with multi-scale feature attention for text classification," in *Proc. IJCAI*, 2018.
[22] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv:1408.5882*. [Online]. Available: https://arxiv.org/abs/1408.5882
[23] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.
[24] A. Conneau, H. Schwenk, and L. Barrault, "Very deep convolutional networks for text classification," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, vol. 1, 2017, pp. 1107–1116.
[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process.*, 2017, pp. 6000–6010.
[26] S. Lai, L. Xu, and K. Liu, "Recurrent convolutional neural networks for text classification," in *Proc. AAAI*, vol. 333, 2015, pp. 2267–2273.
[27] A. Vaswani, N. Shazeer, and N. Parmar, "Attention is all you need," 2017, *arXiv:1706.03762*. [Online]. Available: https://arxiv.org/abs/1706.03762
[28] Z. Yang, D. Yang, and C. Dyer, "Hierarchical attention networks for document classification," in *Proc. HLT-NAACL*, 2016, pp. 1480–1489.
[29] K. Sheng, R. Socher, and D. C. Manning, "Improved semantic representations from tree-structured long short-term memory networks," 2015, *arXiv:1503.00075*. [Online]. Available: https://arxiv.org/abs/1503.00075
[30] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. NAACL*, 2018.
[31] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018.
[32] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," OpenAI, Tech. Rep., 2018.
[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, 2018.
[34] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, "Tutorial on learning from multi-label data," in *Proc. ECML PKDD*, Bled, Slovenia, 2009.
[35] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognit.*, vol. 37, no. 9, pp. 1757–1771, Sep. 2004.
[36] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Machine Learning and Knowledge Discovery in Databases* (Lecture Notes in Artificial Intelligence), vol. 5782, W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, Eds. Berlin, Germany: Springer, 2009, pp. 254–269.
[37] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Mach. Learn.*, vol. 73, no. 2, pp. 133–153, 2008.
[38] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Machine Learning* (Lecture Notes in Computer Science), vol. 4701, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Germany: Springer, 2007, pp. 406–417.
[39] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.
[40] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Principles of Data Mining and Knowledge Discovery* (Lecture Notes in Computer Science), vol. 2168, L. De Raedt and A. Siebes, Eds. Berlin, Germany: Springer, 2001, pp. 42–53.
[41] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in Neural Information Processing Systems*, vol. 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA, USA: MIT Press, 2002, pp. 681–687.
[42] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proc. 22nd Annu. Int. SIGIR*, 1999.
[43] D. D. Lewis, Y. Yang, and T. G. Rose, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, Apr. 2004.
[44] G. Madjarov, D. Kocev, and D. Gjorgjevikj, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognit.*, vol. 45, no. 9, pp. 3084–3104, 2012.
[45] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
[46] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[47] L. Zhang, Y. Zhu, and W. X. Zheng, "Synchronization and state estimation of a class of hierarchical hybrid neural networks with time-varying delays," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 459–470, Feb. 2015.
[48] L. Zhang, Y. Zhu, and W. X. Zheng, "State estimation of discrete-time switched neural networks with multiple communication channels," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 1028–1040, Apr. 2017.
[49] K. Kowsari, M. K. Jafari, and M. Heidarysafa, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

JIAQI YAO received the bachelor's degree in electronic engineering from Tsinghua University, in 2014. He is currently pursuing the Ph.D. degree with the National Key Laboratory of Science and Technology on Blind Signal Processing. His research interests are in machine learning and natural language processing.

KEREN WANG received the Ph.D. degree from the National Key Laboratory of Science and Technology on Blind Signal Processing, in 2014. His research interests are in machine learning video steganography and natural language processing.

JIKUN YAN received the Ph.D. degree from the National Key Laboratory of Science and Technology on Blind Signal Processing, in 2008. His research interests are in machine learning and natural language processing.