Ensemble Application of Convolutional and Recurrent Neural Networks for Multi-label Text Categorization

Guibin Chen[1], Deheng Ye[1], Erik Cambria[1], Jieshan Chen[2], Zhenchang Xing[3]
[1] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2] School of Mathematics, Sun Yat-sen University, China
[3] Research School of Computer Science, Australian National University, Australia
gbchen@ntu.edu.sg, ye0014ng@e.ntu.edu.sg, cambria@ntu.edu.sg, chenjsh28@mail2.sysu.edu.cn, zhenchang.xing@anu.edu.au

***Abstract*—Text categorization, or text classification, is one of key tasks for representing the semantic information of a document. Multi-label text categorization is finer-grained approach to text categorization which consists of assigning multiple target labels to documents. It is more challenging compared to the task of multi-class text categorization due to the exponential growth of label combinations. Existing approaches to multi-label text categorization fall short to extract local semantic information and model label correlations. In this paper, we propose an ensemble application of convolutional and recurrent neural networks to capture both the global and the local semantics of texts and to model high-order label correlations while having a tractable computational complexity. Extensive experiments show that our approach achieves the state-of-the-art performance when the CNN-RNN model is trained using a large-sized dataset.**

I. Introduction

Text categorization is a multi-label classification problem in the context of text processing. It refers to the task of assigning one or multiple categories (or labels) to text, which is a time-consuming and sometimes intractable task. Text categorization has been enormously applied to real-world problems, e.g., information retrieval [1], affective computing [2], sentiment analysis [3], email spam detection [4], multimodal content analysis [5], and more.

Efficient ways to tackling this task require advanced semantic representations of texts that go beyond the bag-of-words model as this ignores local orderings of word and, hence, is unable to grasp local semantic information [6]. Moreover, labels often exhibit strong co-occurrence dependences. For example, a color label green often occurs with the subject labels tree or leaf, however, the color label blue will never come with the subject label dog. By extracting such label correlations instead of considering labels independently, a more efficient prediction model is expected to be obtained.

In this work, we propose an ensemble application of convolutional neural network (CNN) and recurrent neural network (RNN) to tackle the problem of multi-label text categorization. We develop a CNN-RNN architecture to model the global and local semantic information of texts, and we then utilize such label correlations for prediction.

In particular, we employ the state-of-the-art word-vector based CNN feature extraction and deal with high-order label correlation while keeping a tractable computational complexity by using RNN. We perform experiments to investigate the influence of parameters in our model. Additionally, we compare our method with several baselines using two publicly available datasets, i.e., Reuters-21578 and RCV1-v2. The former reflects the behavior of our model in a relatively small dataset, and the latter is considered as a large-scale dataset.

The remainder of the paper is organized as follows: Section II briefly introduces the basic concepts for multi-label text categorization; we review related work in Section III; Section IV presents the details of our CNN-RNN method; experiments for exploring the parameters of our model and comparisons with the baselines are illustrated from Section V to Section VII; finally, Section VIII concludes the paper.

II. Background

Multi-label text categorization can generally be divided into two sub-tasks: text feature extraction and multi-label classification. An introduction of these two sub-tasks will be described in the following two subsections.

A. Text features

Raw text information cannot be directly used in the subsequent multi-label classifier. Many research works represent the features of texts and words as vectors in a framework termed vector space model (VSM). Instead of representing the whole text in one step, e.g., tf-idf weighting for a document, most recent works focus on the distributional representation of individual words, which is nevertheless pre-processed and tokenized from the original raw text data. Grouping similar words in the vector space has shown good performance in many natural language processing (NLP) tasks, e.g., multi-task learning [7], sentence classification [8], sentiment analysis [9], semantically equivalent detection [10], etc. Among them, research efforts starting from [11] bring in a word2vec model that can capture both syntactic and semantic information with demonstrated effectiveness in many NLP tasks.

The basic idea of the word2vec model in [11] is to consider the contextual words as a prediction given a center word, which utilizes the local context of words to get its hidden semantics. This vector representation of word vectors will be used in our word-vector based CNN as the initial representation of the raw text.

B. Multi-label classification

Since text features can be well represented as a feature vector, the next step is to do multi-label classification for such features. Each input instance is a high dimensional feature vector $X \in R^m$, which is assigned to a subset y of the label space Y with L possible labels. The task is to learn a function h: $R^m \rightarrow 2^Y$ from the training data $D = \{(x_i, y_i)|1 \leq i \leq N\}$, where N is total amount of training data, x_i is the i-th instance input feature vector in R^m and $y_i \subset Y$ is a subset of the label space Y. Thus, unlike traditional supervised classification problems where each instance is only assigned with one label, the generalization to the multi-label problem presents multi-label assignments for each instance simultaneously. The exponential growth of possible label combinations, however, makes multi-label classification a very challenging problem.

III. Related Work

Since text categorization consists of two subtasks, namely text feature extraction and multi-label classification, we will discuss some existing works for these two subtasks. Many algorithms have been proposed for representing the features of text. Previous works use either tf-idf weighting [12] or n-gram information [13] for the whole document or local sequence of words. However, tf-idf ignores the local ordering of words and loses some semantic information, while incorporating more n-gram information is not effective due to the data sparsity. Thus, other research works [14], [15], [16], [11] proposed to learn a low-dimensional vector representation of words considering its local context, which gain successful use in many NLP tasks.

On the other hand, recent developments in multi-label classification algorithm fall into two main categories: problem adaption and algorithm adaption. Problem adaption is about transforming multi-label classification into some simpler problems. For example, [17] tracts each label as individual classification so that any binary classification algorithm can be adapted. Classifier chains [18] are designed to model the high-order correlation between labels by transforming the multi-label problem into a chain of binary classification problems, where the subsequent classifiers in the chain will take into account the preceding predictions. For algorithm adaptation, ML-KNN [19], ML-DT [20], and Rank-SVM [21] adapt state-of-the-art machine learning algorithms, e.g., k-nearest-neighbors, decision trees, support vector machines, to solve multi-label classification problems. Most methods, however, can only to capture the first or second order label correlation or are computational intractable in considering high-order label correlations.

Recently, several deep learning works have applied neural networks, such as CNN and RNN, to various NLP tasks, including multi-task learning [7], sentence classification [8], multimodal analysis [23], semantically equivalent detection [10], polarity detection [24], machine translation [22], sarcasm detection [25], aspect extraction [27], personality detection [26], and more. As for multi-label text categorization, several neural-network based methods have been proposed. [28] proposes a BP-MLL that utilizes a full-connected neural network and a pairwise ranking loss function to tackle this task. [29] claims that the pairwise ranking loss is not consistent with the rank loss, and introduces the cross-entropy loss instead. Another more recent neural network method, called ML-HARAM, is introduced in [30]. Unlike CNN and RNN, however, these types of neural networks are still not efficient enough to tackle high dimensional and large-scale data. For this reason, we introduce a CNN-RNN combined method for multi-label text categorization.

IV. CNN-RNN Multi-Label Text Categorization

In this section, the details of the proposed CNN-RNN method are introduced. Our approach consists of two parts: the CNN part for extracting text features and the RNN part for multi-label prediction. Before training the whole CNN-RNN model, we pre-train a word2vec model that can capture the local features for each word and fix it in the following supervised CNN-RNN training. CNN is used to extract a global fix-length feature vector for the input text. Then, with these feature vectors, the "initial state" or prior knowledge of the RNN is determined and used to predict a sequence of labels. The RNN prediction can model the high-order correlation between labels so that it is expected to improve the performance over the predictors that model the lower-order labels correlation.

A. Word-vector based CNN

The way to use CNN to extract text features is similar to the one used in [8]. The details are as follows: the input text is firstly preprocessed and tokenized into a sequence of words. Each word will look up the word embedding matrix obtained from the word2vec model as described in Section II-A. The original text is then a concatenation of word vectors $w_i \in R^k$. Thus, a text of length m is represented as

$$S = w_{1:m} = w_1 \oplus w_2 \oplus ... \oplus w_m \tag{1}$$

where $\oplus$ is the vector concatenation operator. We can treat the sentence vector as "image", and perform convolution on it via linear filters. In text applications, because each word is represented as a k-dimensional vector, it is reasonable to use filters with widths equal to the dimensionality of the word vectors (i.e., k). Thus we simply vary the *window size* (or *height*) of the filter, i.e., the number of adjacent words considered jointly. Let $w_{i:i+h-1}$ refer to the concatenation vector of h adjacent words $w_i, w_{i+1}, ..., w_{i+h-1}$. A convolution operation involves a filter $w \in \mathbb{R}^{hk}$ (a vector of $h \times k$ dimensions) and a bias term $b \in \mathbb{R}^h$, which is applied to h words to produce a new value $o_i \in \mathbb{R}$:

$$o_i = w^T \cdot x_{i:i+h-1} + b \tag{2}$$

where $i = 1...n - h + 1$, and $\cdot$ is the dot product between the filter vector and the word vector. This filter is applied repeatedly to each possible window of h words in the sentence (i.e., $x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}$) to produce an *output sequence* $o \in \mathbb{R}^{n-h+1}$, i.e., $o = [o_1, o_2, ..., o_{n-h+1}]$. We apply a non-linear *activation function* f to each o_i to produce a *feature map* $c \in \mathbb{R}^{n-h+1}$ where $c_i = f(o_i)$. The non-linear activation functions is chosen as $ReLu$, as shown in [31], as this function have more benefits over the $sigmoid$ and $tanh$ one.

$$ReLu(o_i) = max(0, o_i) \tag{3}$$

One may also specify multiple kinds of filters with different window sizes, or use multiple filters for the same window size to learn complementary features from the same word windows. The dimensionality of the feature map generated by each filter will vary as a function of the text length and the filter's window size. Thus, a pooling function is then applied to each feature map to induce a fixed-length vector. A common strategy is *1-max pooling*, which extracts a scalar (i.e., a feature vector of length 1) with the maximum value for each filter.

As a result, a filter with a certain window size will only produce one scalar value, which means that if there are i window-size filters and each of them have j filters, there will be $i \times j$ outputs in total. For example, concatenating output of k filters of each the three window-size (1, 3, 5) will result in a 3k-dimension output vector. By projecting its output vector into a lower dimensional space using a full-connect layer, we obtain a low-dimension feature representation for the text. This feature vector can be fed into the next layer of the CNN for further convolution, or be used as the output vector for different NLP tasks. Here, this text feature vector is used as the input for the RNN.

B. Label sequence prediction by RNN

RNN is mainly used for sequential labeling or prediction, such as language model, auto-encoder, etc. It can be considered as a extension of the hidden Markov model that introduces nonlinearity transitions to model long-term nonlinear dependences. LSTM is considered as one of most successful RNN variants, which introduces three additional gates. In text domain, LSTMs have been involved in the task of sentiment analysis [32], sentence classification [33], etc. While using it for the whole long document, the training of LSTMs is not stable and underperformed traditional linear predictors as shown in [34]. Moreover, the training and testing time of LSTMs are also time/resource consuming for the long document. [34] illustrates that the training of LSTM can be stabilized by pre-training the LSTMs as sequence auto-encoders or recurrent language models. However, the above problem is avoided when we use LSTMs for label sequence predictions, which is typically much shorter than a document. The label sequence here is the assignments of ordered labels to a text. Although several variants of LSTMs exists, the standard LSTM is used. An additional word embedding layer is also applied for the labels.

A LSTM consists of three gates: an input gate i, an output gate o and a forget gate f. The three gates work collaboratively to control what to be read by input, what to output, and what should be forgotten, so that some complex long-term relationship can be modeled. The standard LSTM is expressed as follows:

$$\begin{aligned} i &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}), \\ o &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}), \\ f &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}), \\ u &= \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}), \\ c_t &= i \odot u + f \odot c_{t-1}, \\ h_t &= o \odot \tanh(c_t) \end{aligned} \tag{4}$$

where $\odot$ denotes element-wise multiplication and $\sigma(.)$ is the sigmoid function. $x_t \in R^d$ is the input from lower layer at time step t, and d can be the dimension of word vector of the labels if the lower layer is word embedding of the labels or can be the hidden-state dimension of the lower layer if the lower layer is LSTM. If there are q LSTM units, then $h_t \in R^q$, $W^{(.)} \in R^{q \times d}$, $U^{(.)} \in R^{q \times q}$ and $b^{(.)} \in R^q$ for all types (i, o, f, u). Memory cell c_t is the key in LSTM which keeps the long-term dependencies while getting rid of the vanishing/exploding gradients problems. Forget gate f is used to erase some parts of memory cells, while the input gate i and output gate o control what to read into and write out of from memory.

C. Combine CNN and RNN

As shown in Section IV-A, the CNN is used to extract the feature of text so that the semantic information is represented as a output vector from CNN. After feeding this output vector into LSTM as the initial state for the label prediction, the whole network is able to predict a sequence of related labels to the input documents according to the features extracted by CNN (Figure 1. The feeding of the feature vector into the LSTM is by a linear transformation as an additional addition term $W^{(T)}T$ in (4) for each types (i, o, f, u), where T is the output text feature from CNN with a fix-dimension t, $W^{(T)} \in R^{q \times t}$, and q is the hidden dimension of LSTM. For example, the formula for the input gate is changed to

$$i = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)} + W^{(T)}T) \tag{5}$$

The label sequence prediction is always started with the token $< START >$. In each time step, there is softmax layer above the top layer of LSTM to calculate the probability of each label by using linear transform the hidden-state of top LSTM layer first. Then, the label with maximal probability will be taken a prediction. The prediction of labels will be ended with the $< END >$ token. Thus, for each text, a various length of label sequence will be predicted. The ideal case will be that label sequence of each text is exactly matched to the subset of labels that belongs to that text. For example, in Figure 1, the input text is "the cat is sleeping near a small American flag in a bed" and this corresponding ground truth labels include the subject (Animal), the location (United States) and the time (Midnight).

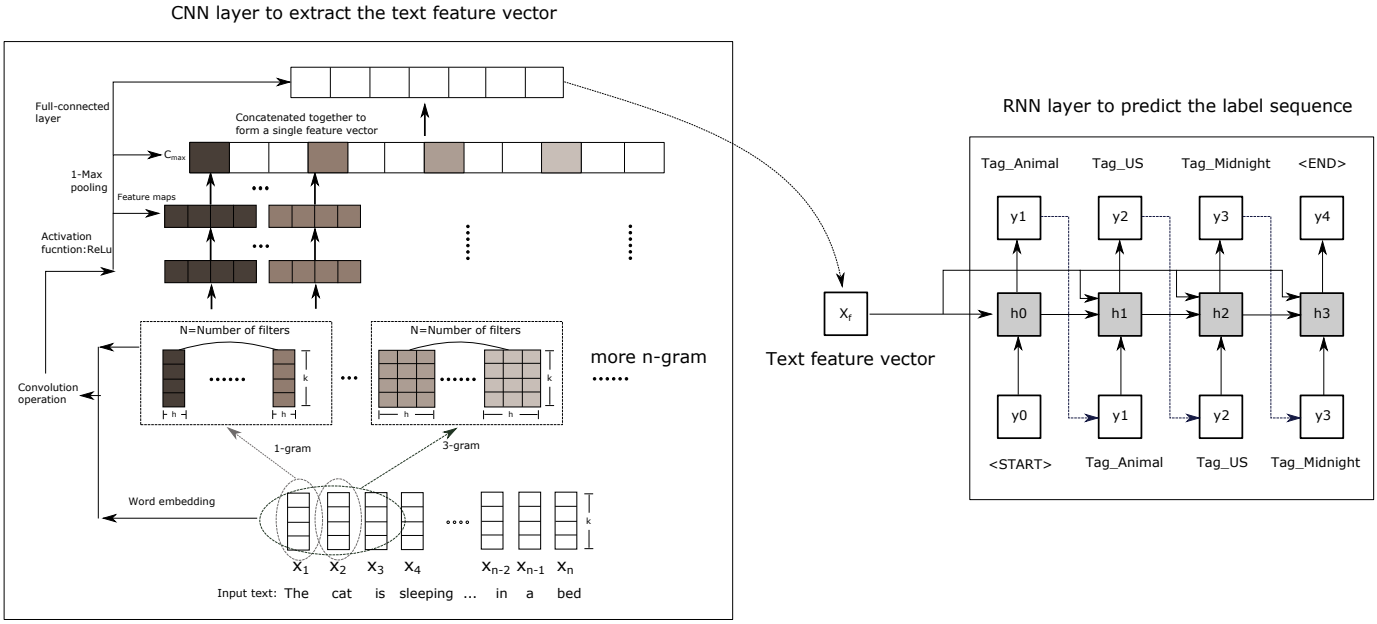


Fig. 1: Our Proposed CNN-RNN Model

D. Training

Two steps of trainings are performed. The first training step is to train a word2vec model by using all the tokenized texts as unlabelled data. Then, this word2vec model is fixed in the second training step, that is the supervised training for the CNN-RNN model. Using the softmax classifier as the upper layer of LSTM for labels prediction, the cross-entropy loss is then back-propagated from RNN down to CNN to update the weights of both the CNN-RNN model. One state-of-the-art update policy Adam [35] is chosen instead of stochastic gradient decent (SGD) for a faster convergence. Besides, for regularization, we apply $l2$-constraints over all weights in CNN and RNN and use dropout with rate 0.5 in CNN part.

V. Datasets

To test the performance of our algorithm, we use the two publicly-available dataset, Reuters-21578 and RCV1-v2.

- **Reuters-21578:** The documents in this dataset were collected from the Reuters newswire in 1987. This was once a popular dataset for researchers who worked in text categorization since 1996 (now superseded by RCV1). Adapted from this preview Reuters-22173 version, it now had 21,578 documents. In the Modified Apte ("ModApte") Split, there are 9,603 docs for training and 3,299 docs for testing. We use this split to do the empirical studies of parameters of our network. After that, we use all these documents for the ten-fold cross-validation to compare various algorithms.
- **RCV1-v2:** Reuters Corpus Volume I (RCV1) is an archive of over 800,000 manually categorized newswire stories recently made available by Reuters, Ltd. for research purposes. By correcting the errors of the raw RCV1 data, a text categorization test collection is produced which is called RCV1-v2. As in the LYRL2004 train/test split in [36], there are a training set (document IDs 2286 to 26150) and test set document IDs 26151 to 810596). We only consider the topic categories as labels whose amount is 103 and also collect all these dataset to do ten-fold cross-validation instead of original split.

TABLE I: Dataset Overview. M is the number of documents, D is the total vocabulary size, L is the number of labels, C is the average number of labels per document

dataset	M	D	L	C
Reuters-21578	10789	18637	90	1.13
RCV1-v2	804414	47236	103	3.24

The characteristics of the above dataset are summarized in Table I. Since RCV1-v2 dataset provides both the token and vector (tf-idf, cosine-normalized) versions, we can directly use the token version for our CNN-RNN model and the vector model for all the baseline algorithms. However, the Reuters-21578 only has a raw dataset. Thus, we preprocess these documents and choose the top 1000 tf-idf features according to their document frequency. Then, each document is presented as a 1000-dimensional feature vector. This vector model can then be used for all baseline algorithms.

VI. Evaluation Metrics

The evaluation metrics of multi-label classification have two types: ranking and classification. Since our model will not provide a complete ranking for all the labels, the ranking metrics evaluated here will include one-error. For the classification metrics, we will consider the hamming loss, the macro/micro-averaged precision, recall, and F1 score.

- The **one-error** counts the fraction of instances whose top-1 predicted label is not in the relevant labels.
- The **hamming loss** counts the symmetric difference of the predicted labels and the relevant labels and calculates the fraction of its difference over the label space.
- **Precision**, **Recall** and **F1 score** are binary evaluation measures $B(tp, tn, fp, fn)$ to evaluate the performance of a classification problem, which is calculated based on the number of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). There are two ways to calculate these metrics over the whole test data: macro-averaged and micro-averaged. Macro-averaged refers to the average performance (Precision, Recall and F1 score) over labels, while micro-averaged counts all true positives, true negatives, false positives and false negatives first among all labels and then has a binary evaluation for its overall counts.

VII. Experimental Results

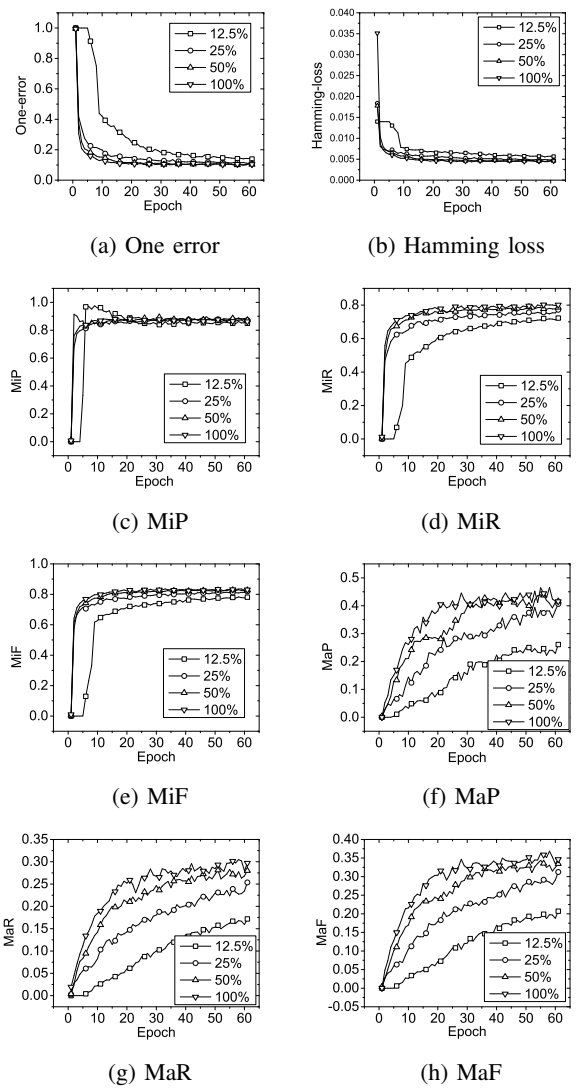
In this section, we perform experiments to investigate the influence of complexity of the network on the performance and also compare our method with several baselines. For the parameter studies, we mainly focus on the parameters of the RNN part, since the investigation of CNN structure is already shown in [8]. Here, the structure of the CNN consists of a pre-trained word embedding layer and one layer CNN with several window sizes. The parameters of CNN is chosen based on the validation results for the Reuters-21578 data, such that it will not limit the overall performance when incorporating RNN.

The window size is chosen up to 9-gram. The dimension of word vector is chosen as 400 while the number of filters in each window size is chosen as 128. The nonlinear function we choose is $Relu$, because [31] shows that a rectified linear unit has more benefits than $sigmoid$ and $tanh$ function. 1-Max-pooling is applied in all filters so that each window size will output a 128-dimension feature vector. Besides, we concatenate these feature and use a full-connection layer to project it into a 256-dimension output vector, which is then used as the input or the prior knowledge of RNN.

In the RNN part, the parameters include the dimension of label embedding, the dimension of hidden state and also the number of layer of LSTM. An empirical study over these parameters are performed by using the Reuters-21578 data with Modified Apte ("ModApte") Split. Apart from the investigation of the network itself, we also compare its results with the several baseline: Binary Relevance (BR), Classifier Chain (CC), ML-kNN, and ML-HARAM.

A. Complexity of RNN versus performance

The parameters of RNN include the dimension of labels, the hidden dimension and the number of layers. More layers will generally be able to get higher abstract content and keep more long-term memory. Here, to be simplistic, we focus one layer LSTM parameter investigation without the loss of generalization.



(a) One error (b) Hamming loss

(c) MiP (d) MiR

(e) MiF (f) MaP

(g) MaR (h) MaF

Fig. 2: Metrics versus training epoch for various RNN complexity

We simultaneously set the dimension of labels and hidden dimension to be α of the input dimension from CNN, that is the dimension of the feature vector. We monotonically increase α from 12.5% to 100% with the doubling step. We observe the various performances over the validation set of the Reuters-21578 data with Modified Apte ("ModApte") Split. Then the various performances versus training epoch for the different RNN complexity is shown in Figure 2. Different curves correspond to different α values.

As shown in the figures, when the ratio is set to 50%, performance is superior than the lower ratio ones and comparable with the 100% case. Besides, when the training epoch goes beyond 50, performance does not change significantly and begin to converge. Thus, to save the computational effort and keep a comparable performance, in the experiment with baselines, we set the CNN-RNN parameter ratio as 50% and the training epoch is limited to 50.

TABLE II: Comparison of various algorithms (Mean $\pm$ Deviation)

Algorithm	BR	CC	ML-kNN	ML-HARAM	CNN-RNN
Reuters-21578					
One-error	0.091 ± 0.012	0.084 ± 0.010	0.474 ± 0.031	0.112 ± 0.024	$\mathbf{0.083 \pm 0.009}$
H-loss	0.0032 ± 0.0005	$\mathbf{0.0031 \pm 0.0005}$	0.0088 ± 0.0009	0.0066 ± 0.0011	0.0038 ± 0.0004
MiP	$\mathbf{0.940 \pm 0.014}$	0.937 ± 0.014	0.803 ± 0.041	0.748 ± 0.030	0.902 ± 0.014
MiR	0.823 ± 0.019	$\mathbf{0.828 \pm 0.019}$	0.473 ± 0.032	0.775 ± 0.036	0.813 ± 0.026
MiF	0.878 ± 0.016	$\mathbf{0.879 \pm 0.016}$	0.595 ± 0.035	0.762 ± 0.032	0.855 ± 0.015
MaP	0.464 ± 0.038	$\mathbf{0.470 \pm 0.046}$	0.342 ± 0.032	0.298 ± 0.040	0.369 ± 0.002
MaR	0.350 ± 0.020	$\mathbf{0.361 \pm 0.023}$	0.205 ± 0.028	0.242 ± 0.019	0.287 ± 0.026
MaF	0.385 ± 0.024	$\mathbf{0.395 \pm 0.029}$	0.239 ± 0.027	0.237 ± 0.023	0.322 ± 0.017
RCV1-v2					
One-error	0.012 ± 0.001	0.045 ± 0.002	N/A	N/A	$\mathbf{0.010 \pm 0.002}$
H-loss	0.0088 ± 0.0003	0.0093 ± 0.0004	N/A	N/A	$\mathbf{0.0086 \pm 0.0004}$
MiP	$\mathbf{0.898 \pm 0.005}$	0.886 ± 0.005	N/A	N/A	0.889 ± 0.010
MiR	0.812 ± 0.008	0.811 ± 0.008	N/A	N/A	$\mathbf{0.815 \pm 0.015}$
MiF	$\mathbf{0.853 \pm 0.006}$	0.847 ± 0.006	N/A	N/A	0.849 ± 0.012
MaP	0.794 ± 0.006	0.768 ± 0.004	N/A	N/A	$\mathbf{0.803 \pm 0.004}$
MaR	0.623 ± 0.008	0.644 ± 0.009	N/A	N/A	$\mathbf{0.646 \pm 0.012}$
MaF	0.687 ± 0.008	0.693 ± 0.007	N/A	N/A	$\mathbf{0.712 \pm 0.008}$

B. Baseline comparison

We test various algorithms over the above datasets, including Binary relevance (BR), Classifier chain (CC), ML-kNN, ML-HARAM and our CNN-RNN model. BR is used to represent multi-label algorithms without taking into account the correlation between labels, CC is an algorithm that takes into consideration the high-order label correlation, ML-kNN is a transformation-based algorithm, and ML-HARAM is a state-of-the-art neural network based algorithm.

Ten-fold cross-validation is performed over these algorithms. Each of the two datasets is randomly divided into 10 equal size subsets. We perform ten iteration experiments, and each of them takes 9 subsets, and the rest one is hold out for testing. Then, we calculate the mean and deviation of the 10 experiments. The result is shown in Table II. The implementation of all the baseline methods are done in Scikit-multilearn [37], an open-source library for the multi-label classification that is built on top of the scikit-learn ecosystem [38]. In BR and CC, the linear SVMs are used as the base classifier. We investigate the number of neighbors in ML-kNN from 5 to 20 with the step of 5, and choose 10 finally that is optimized for both Hamming-loss and F1 score. ML-HARAM follows its default setting.

As shown in Table II, for the dataset Reuters-21578, the CNN-RNN model does not shown an improvement over the two traditional methods (BR and CC). We investigate the performance over training data, and find that this model tends to overfit the training data of this relative small dataset. For example, the micro-averaged precision in the training dataset is 99%. Besides, CC only has a slightly better performance than BR except for the micro-averaged precision.

The reason could be that the average number of labels per document in the Reuters-21578 are about one, so that it makes no much sense to model high-order correlation. Notice that ML-kNN is inferior to any other methods. The state-of-art neural network based approach ML-HARAM is also inferior to our proposed neural network based approach in all metrics. The deep learning method will generally benefit from a vast number of dataset. For the RCV1-v2 dataset, the proposed method outperforms any other baseline methods in most of the metrics, especially in macro-averaged metrics. We conjecture that modeling the high-order label correlation helps to predict the minor labels that occurs less frequently in the dataset.

Meanwhile, both of the traditional methods, BR and CC, show a substantial decreased performances in micro-averaged metrics and the hamming loss. The increases in the Macro-averaged metrics relative to the previous dataset are due to the more balanced distribution of labels in this dataset. Notice that the metrics for ML-kNN and ML-HARAM are not available as the Scikit-multilearn implementation of these algorithms is not scalable with the data size and results in memory error. We expect that our model can also outperform these two methods as it does in the small dataset. However, further experiments are required to confirm it.

VIII. Conclusion

Multi-label text categorization is the task of assigning predefined categories to textual documents. Existing approaches to multi-label text categorization fall short to extract local semantic information and to model label correlations. In this paper, we proposed a convolutional neural network (CNN) and recurrent neural network (RNN) based method that is capable of efficiently representing textual features and modeling high-order label correlation with a reasonable computational complexity.

Our evaluations reveal that the power of the proposed method is affected by the size of the training dataset. If the data size is too small, the system may suffer from overfitting. However, when trained over a large-scale dataset, the proposed model can achieve the state-of-the-art performance.

REFERENCES

[1] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik, "A multi-view embedding space for modeling internet images, tags, and their semantics," *International journal of computer vision*, vol. 106, no. 2, pp. 210–233, 2014.

[2] S. Poria, E. Cambria, R. Bajpai, and A. Hussain, "A review of affective computing: From unimodal analysis to multimodal fusion," *Information Fusion*, vol. 37, pp. 98–125, 2017.

[3] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 102–107, 2016.

[4] X. Carreras and L. Marquez, "Boosting trees for anti-spam email filtering," in *RANLP*, 2001, pp. 58–64.

[5] S. Poria, I. Chaturvedi, E. Cambria, and A. Hussain, "Convolutional MKL based multimodal emotion recognition and sentiment analysis," in *ICDM*, Barcelona, 2016, pp. 439–448.

[6] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research," *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, 2014.

[7] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[8] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[9] E. Cambria, J. Fu, F. Bisio, and S. Poria, "AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis," in *AAAI*, Austin, 2015, pp. 508–514.

[10] D. Bogdanova, C. dos Santos, L. Barbosa, and B. Zadrozny, "Detecting semantically equivalent questions in online user forums," *CoNLL 2015*, p. 123, 2015.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[12] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[13] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, no. 8, pp. 1157–1166, 1997.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[15] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[17] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[18] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 254–269.

[19] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[20] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 42–53.

[21] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in neural information processing systems*, 2001, pp. 681–687.

[22] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[23] S. Poria, E. Cambria, and A. Gelbukh, "Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis," in *EMNLP*, 2015, pp. 2539–2544.

[24] I. Chaturvedi, Y.-S. Ong, I. Tsang, R. Welsch, and E. Cambria, "Learning word dependencies in text by means of a deep recurrent belief network," *Knowledge-Based Systems*, vol. 108, pp. 144–154, 2016.

[25] S. Poria, E. Cambria, D. Hazarika, and P. Vij, "A deeper look into sarcastic tweets using deep convolutional neural networks," in *COLING*, 2016, pp. 1601–1612.

[26] N. Majumder, S. Poria, A. Gelbukh, and E. Cambria, "Deep learning based document modeling for personality detection from text," *IEEE Intelligent Systems*, vol. 32, no. 2, 2017.

[27] S. Poria, E. Cambria, and A. Gelbukh, "Aspect extraction for opinion mining with a deep convolutional neural network," *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016.

[28] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[29] J. Nam, J. Kim, E. L. Mencía, I. Gurevych, and J. Fürnkranz, "Large-scale multi-label text classification—revisiting neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 437–452.

[30] F. Benites and E. Sapozhnikova, "Haram: a hierarchical aram neural network for large-scale text classification," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 847–854.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[32] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1422–1432.

[33] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[34] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in Neural Information Processing Systems*, 2015, pp. 3079–3087.

[35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of machine learning research*, vol. 5, no. Apr, pp. 361–397, 2004.

[37] P. Szymański, "Scikit-multilearn: Enhancing multi-label classification in python," 2014, manuscript in preparation.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.