Multi-Label Reasoner for Text Classification

Anonymous EMNLP-IJCNLP submission

Abstract

Multi-label classification is a common and important task in natural language processing. Existing methods either ignore label correlations, or use chained classifiers to handle labels that are not necessarily sequential. In this work, we propose Multi-Label Reasoner (ML-Reasoner), a reasoning-based algorithm for the task of multi-label classification. The ML-Reasoner introduces a novel iterative reasoning mechanism to effectively utilize the inter-label information, where each instance of reasoning in the prediction process takes the previous reasoning results as additional input. Results from extensive experiments show that our methods outperform existing state-of-the-art techniques. Further analyses of our experiment results verify the effectiveness of reasoning for multi-label classification, and additionally reveal some reasons for why it is so effective. [1]

1 Introduction

Multi-Label Classification (MLC) assigns multiple non-exclusive labels to each instance. This is applicable to classification for many scenarios in natural language processing, such as for papers covering multiple disciplines (Read et al., 2011; Yang et al., 2018), blog posts discussing multiple topics (Tsoumakas et al., 2008; Katakis et al., 2008), comments revealing multiple emotions (Herrera et al., 2016), etc. Therefore, MLC plays an important and fundamental role in natural language processing, and has been extensively studied (Klimt and Yang, 2004; Gopal and Yang, 2010; Yang and Gopal, 2012).

The Binary Relevance (BR) algorithm (Gonçalves and Quaresma, 2003) transforms the MLC problem into multiple independent binary classification problems.

[1] The data and code are available at https://anonymous.

Text	This paper is about multi-label classifications for text.
relevant labels	machine learning, linguistics, natural language processing
irrelevant labels	math

Table 1: An example of multi-label classification for text. In this case, the label "natural language processing" is dependent on both "machine learning" and "linguistics". This demonstrates two key features of label relationships in the task of MLC: first, the labels are **non-exclusive**, and, second, the dependencies between labels are **not necessarily sequential**.

However, this method completely ignores the relevant information between labels. Most researchers (Galar et al., 2011; Dembszynski et al., 2010; Tenenboim-Chekina et al., 2010; Zhang and Zhang, 2010) have pointed out that it is beneficial, *even necessary*, to make use of the inter-label relationships for the task of MLC.

To address this issue, the Label Powerset (LP) proposed in Boutell et al. (2004) treats each potential label combination as a unique class identifier. The LP method implicitly accounts for the hidden relationships among labels, because sets of labels with a certain correlation tend to generate the same or similar label combinations (Herrera et al., 2016). But it suffers from a number of problems. First, it is only able to deal with label combinations that were encountered in the training set. Second, the enlarged solution space makes it difficult to train a classifier, a drawback which negates the advantage gained by the implicit consideration for inter-label dependencies.

Alternatively, in order to model label correlations, classifier chains (CCs) (Read et al., 2011) link together multiple binary classifiers, with each classifier using the prediction from the previous one as its input. Similarly, Yang et al. (2018)

have approached the MLC task by viewing it as a sequence-to-sequence task; this approach is named Sequence Generation Model (SGM). The major difference between SGMs and CCs is that SGMs use neural-network based multi-class classifiers with shared parameters to replace the binary ones in CCs. However, these approaches all **ignore** the fact that relationship between labels is not necessarily sequential. Taking Table 1 as an example, the label "natural language processing" is related to *both* "machine learning" and "linguistics". Even worse, those algorithms are also **highly sensitive** to the arrangement of classifiers (Herrera et al., 2016) or sorting of labels (Yang et al., 2018).

To tackle the above issues, we propose *Multi-Label Reasoner* (ML-Reasoner), a reasoning-based algorithm for the task of MLC. We employ multiple binary classifiers to predict each label directly instead of viewing each possible label combination as a category. This enables us to avoid the potentially huge solution space and make training easier. Importantly, we process all labels at once rather than one label at a time, which is totally different from that of CCs or SGMs. Through this approach, our method avoids chaining classifiers. This enables us to avoid chaining classifiers, worrying about how to arrange the classifiers or sort the labels. In order to make effective use of the label correlations, we introduce a novel iterative reasoning mechanism. During the prediction process, we conduct reasoning on multiple occasions, where each reasoning process takes the previous reasoning result as an input. As such, we can manage to utilize relevant information from the labels while avoiding the shortcomings of previous methods.

The contributions of this paper are as follows:

1. We introduce reasoning for MLC (§ 2) to utilize the correlations between labels and avoid major drawbacks of current methods.

2. We develop a simple but effective model (§ 3), namely ML-Reasoner[2], for handling text multi-label classification tasks.

3. We empirically evaluate ML-Reasoner against several baselines and show that it results in better prediction performance (§ 4). Further analysis (§ 5) highlights the effectiveness of reasoning for handling MLC, and additionally reveals some reasons for why it is so effective.

[2] In order to be concise, we use the term ML-Reasoner to refer to both the algorithm and the model without distinction. We allow the reader to judge what is being referred to through context.

Algorithm 1: Multi-Label Reasoner

Input: Dataset D, Initialized value Z_{init}, Reasoner iteration number T

```
repeat
    foreach instance (x, y) ~ D do
        z = Z_init
        foreach t in 1 ... T do
            z = g(x, z)
        Update the model according to loss(z, y)
until reaching stop criterion
return a multi-label classifier f
```

2 Multi-Label Reasoner Method

We begin by formalizing the MLC. The input space is represented as $\mathcal{X}$; the label space is represented as $\mathcal{L}$, the powerset of $\mathcal{L}$ as $\mathcal{P}(\mathcal{L})$, and the total number of label classes in $\mathcal{L}$ as $k = |\mathcal{L}|$; the output space is represented as $\mathcal{Y}$, including all possible labelsets $Y \in \mathcal{P}(\mathcal{L})$.

The multi-label classifier is essentially the following function: $f : \mathcal{X} \rightarrow \mathcal{Y}$. Let g be the core module of the ML-Reasoner, i.e., the Reasoner, and Z denotes the results produced by g.

To avoid expanding the solution space, reasoners are supposed to deal with each label directly instead of treating label combinations as new categories. In order not to search the optimal arrangement of classifiers and sorting of labels, we set g to process all labels at once rather than one label at a time. Therefore, $Z \in \mathcal{Y}$ and the length of Z is supposed to equal k.

In the interest of making effective use of the label correlations, reasoners should take account of some prior knowledge or useful information between labels. Unfortunately, the prior knowledge is often not readily available. However, we can use previous reasoning results to enhance models. Thus, as well as X, g should also take Z as a input. That is $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Y}$.

Let T be the number of reasoning rounds, i.e. the Reasoner g iteration number. The entire ML-Reasoner takes the following form:

$$f = g(X, g(X, \cdots g(X, Z_{init}\underbrace{)\cdots))}_{T\text{-times}} \quad (1)$$

where Z_{init} is the initialized Z value.

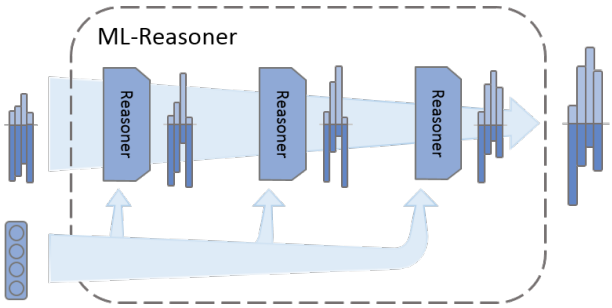

Figure 1: Multi-Label Reasoner method for MLC tasks. This method uses multiple binary classifiers to process all labels at once. The internal *Reasoner* takes account of the previous reasoning results during prediction and it iterates three times. See § 2 for further details.

The ML-Reasoner algorithm is summarized in Alg. 1. The ML-Reasoner method with reasoner iteration number $T = 3$ for MLC tasks is shown in the Fig. 1.

3 Multi-Label Reasoner for Text Classification

In this section, we develop a simple and easy-to-implement model, namely ML-Reasoner, for text multi-label classification.

3.1 Overview

The core of the ML-Reasoner is the design of the Reasoner module. Fig. 2 displays the inner-architecture of the Reasoner as applied by the ML-Reasoner.

3.2 Details

For the word sequence $x = (w_1, \cdots, w_n)$, the ground-truth $y = \{y_1, \cdots, y_k\}$ (where $y_j \in \{0, 1\}$), and given previous reasoning result $Z = \{z_1, \cdots, z_k\}$ (where $z_j \in [0, 1]$), each component in the reasoner is defined as follows.

Text Encoder is responsible for extracting text features. It reads word sequences and produces text representations.

Text Embedding converts the characters in each word into character vectors and then applies a CNN with max-pooling to obtain character-derived embeddings. Each word is also mapped to a pre-trained word vector. Afterwards, the two embeddings are concatenated and passed on to the CNN Encoder:

$$\vec{w}_i = \text{TextEmbedding}(w_i) \in \mathbb{R}^{D_1} \quad (2)$$

where $i \in \{1, \cdots, n\}$.

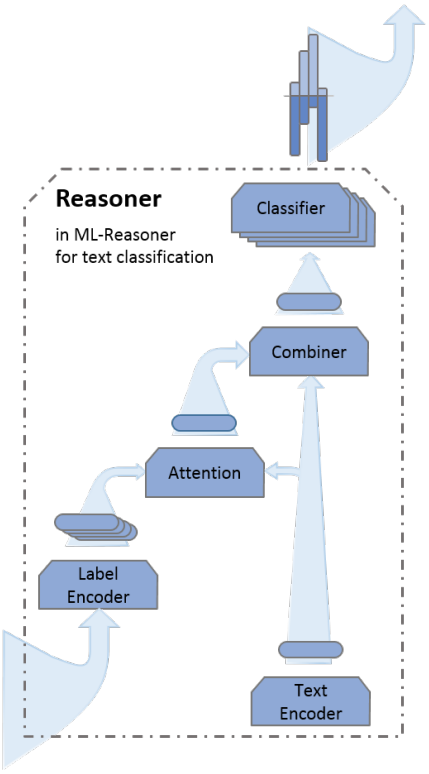

Figure 2: The Reasoner module in ML-Reasoner for text multi-label classification. The Reasoner consist of five components: text encoder, label encoder, attention, combiner, and classifiers. For simplicity, the text representation input to the text encoder is not displayed. The arrow in the lower left corner of this figure denotes the previous reasoning results and the upper right one is the output after reasoning. For more information, see § 3.2.

CNN Encoder (Kim, 2014) with max-pooling can obtain sentence representations efficiently:

$$\vec{x} = \text{CNN-with-maxpool}([\vec{w}_1, \cdots, \vec{w}_n]) \in \mathbb{R}^{D_2} \quad (3)$$

where $[,]$ represents the vector stack operation. Then a linear layer is applied:

$$\vec{x}_{\text{encoded}} = W_1^T \vec{x} + b_1 \in \mathbb{R}^{D_3} \quad (4)$$

where $W_1 \in \mathbb{R}^{D_2 \times D_3}$ and $b_1 \in \mathbb{R}^{D_3}$ are the parameters to be learned.

Label Encoder takes the previous reasoning result Z as its input and outputs a series of label representations.

Label Embedding maps each label into a randomly initialized vector with length D_4:

$$\vec{l}_j = \text{LabelEmbedding}(l_j) \in \mathbb{R}^{D_4} \quad (5)$$

where $j \in \{1, \cdots, k\}$.

Dataset	Total Samples	Label Sets	Words/Sample	Card	Dens
AAPD	55840	54	163.43	2.41	0.045
RCV1-V2	804414	103	123.94	3.24	0.031

Table 2: Summary of the datasets. *Total Samples* and *Label Sets* represent the number of samples and size of the label set respectively. *Words/Sample* is the average number of words per sample. *Card* and *Dens* are the indices introduced in Herrera et al. (2016). *Card* stands for *label cardinality*, which reflects the average number of labels associated with each sample in the dataset. *Dens* stands for *label density*, a normalized value that is calculated by dividing the label cardinality by the total number of labels in the dataset. The dataset AAPD is **more challenging** than dataset RCV1-V2.

Gated Encoder uses the gate-mechanism applied by GRUs (Cho et al., 2014) to combine the label representations $\vec{l}_j$ and the reasoning results $z_j \in \mathbb{R}$:

$$\vec{l}_{j_{\text{encoded}}} = z_j \vec{l}_j \in \mathbb{R}^{D_4} \tag{6}$$

Attention incorporates relevant label information based on text representations. At first, we assign the weight α_j to the j-th label:

$$\alpha_j = \text{softmax}(W_2[\vec{x}_{\text{encoded}}; \vec{l}_{j_{\text{encoded}}}] + b_2) \in \mathbb{R} \tag{7}$$

where $W_2 \in \mathbb{R}^{D_3+D_4}$ and $b_2 \in \mathbb{R}$ are internal parameters of the model and $[;]$ is vector concatenation. Afterwards, the label-aware text vector $\vec{l}_{\text{attention}}$ is calculated:

$$\vec{l}_{\text{attention}} = \sum_{j=1}^{k} \alpha_j \vec{l}_j \in \mathbb{R}^{D_4} \tag{8}$$

Combiner fuses together the information obtained from both the label and text features:

$$\vec{x}_{\text{combined}} = [\vec{x}_{\text{encoded}}; \vec{l}_{\text{attention}}] \in \mathbb{R}^{D_3+D_4} \tag{9}$$

Classifiers with the same structure but different parameters are applied to make predictions for all labels. The j-th classifier is defined as follows:

$$\vec{z}_j = W_{4j}^T \text{tanh}(W_{3j}^T \vec{x}_{\text{combined}} + b_{3j}) + b_{4j} \in \mathbb{R}^2 \tag{10}$$

where $W_{3j} \in \mathbb{R}^{(D_3+D_4)\times D_5}$, $b_{3j} \in \mathbb{R}^{D_5}$, $W_{4j} \in \mathbb{R}^{D_5 \times 2}$ and $b_4 \in \mathbb{R}^2$ are all internal parameters.

3.3 Loss Function

We use *BCELoss* to calculate the loss for the prediction $\vec{z}$ and the ground-truth $\vec{y}$:

$$\ell(\vec{z}, \vec{y}) = \frac{1}{k} \sum_{i=1}^{k} -[y_i \log z_i + (1 - y_i) \log(1 - z_i)] \tag{11}$$

Correspondingly, with the reasoner iteration number set to T, the average loss for the dataset D is defined as follows:

$$loss = \frac{1}{T|D|} \sum_{i=1}^{T} \sum_{j=1}^{|D|} \ell(\vec{z}_{ij}, \vec{y}_{ij}) \tag{12}$$

4 Experiment Settings

In this section, we introduce details regarding the two datasets, baselines, metrics and detailed settings that were used in our experiments.

4.1 Datasets

We conduct experiments on two datasets:

Arxiv Academic Paper Dataset (AAPD) was created by Yang et al. (2018). The task for this dataset is to judge which of the academic disciplines each paper belongs to by the abstracts.

Reuters Corpus Volume I (RCV1-V2) is provided by Lewis et al. (2004). The task for this dataset is to match the desensitized news stories to the relevant topics.

Table 2 shows that the label density of AAPD is *larger*, but that the dataset is *smaller* and the documents are *longer*. Thus, processing the AAPD dataset provides **a greater challenge** than the RCV1-V2 dataset. We follow Yang et al. (2018)'s way of dividing training, validation and test sets. [3]

4.2 Evaluation Metrics

As with previous researchers (Yang et al., 2018; Zhang and Zhou, 2006; Chen et al., 2017), we adopt hamming loss, micro-precision, micro-recall and micro-F_1 as our primary evaluation metrics. We also use precision, recall and F_1 (Herrera et al., 2016) to aid in our analyses.

[3] These data are available online at https://github.com/lancopku/SGM.

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0316	0.644	0.648	0.646
CC	0.0306	0.657	0.651	0.654
LP	0.0312	0.662	0.608	0.634
CNN-RNN	0.0278	0.718	0.618	0.664
SGM	0.0251	0.746	0.659	0.699
SGM+GE	0.0245	0.748	0.675	0.710
ML-Reasoner	**0.0238**	**0.761**	**0.684**	**0.720**

(a) Performance of models on AAPD test set.

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0086	0.904	0.816	0.858
CC	0.0087	0.887	0.828	0.857
LP	0.0087	0.896	0.824	0.858
CNN-RNN	0.0085	0.889	0.825	0.856
SGM	0.0081	0.887	0.850	0.869
SGM+GE	**0.0075**	0.897	**0.860**	0.878
ML-Reasoner	0.0081	**0.912**	0.847	**0.879**

(b) Performance of models on RCV1-V2 test set.

Table 3: Results on both datasets for the baseline and ML-Reasoner models. The term *GE* represents global embedding; *HL*, *P*, *R* and *F1* refer to hamming loss, micro-precision, micro-recall and micro-F_1 respectively. The symbol "-" indicates that the lower the value is, the better the model performs. The symbol "+" is the opposite. The hyperparameter T is set to 2 for each ML-Reasoner. The results of BR, CC, LP, CNN-RNN, SGM and SGM+GE are from (Yang et al., 2018).

Hamming-Loss (Schapire and Singer, 1998) calculates scores according to instance-label pair prediction errors made by the model, with errors being termed as instances of predicting irrelevant labels or failing to predict relevant labels.

Micro-F_1 (Mogotsi, 2010) is calculated after counting the number of true positives, true negatives, false negatives, and false positives. Higher frequency labels usually contribute more to the final measure.

4.3 Baselines

Our model is compared with the following baselines:

BR (Gonçalves and Quaresma, 2003) transforms the MLC task into multiple binary classification tasks.

LP (Boutell et al., 2004) views the MLC task as a multi-class classification problem.

CC (Read et al., 2011) applies a sequence of binary classification tasks to handle MLC tasks.

CNN-RNN (Chen et al., 2017) uses CNNs and RNNs to obtain local and global semantics, and also model the relationships between labels.

SGM (Yang et al., 2018) treats MLC tasks as sequence generation tasks and use seq2seq as a multi-class classifier.

Following the previous work by Chen et al. (2017), a linear SVM is adopted as a base classifier for BR, LP and CC. The ML-Reasoner model is implemented by means of AllenNLP (Gardner et al., 2017), an open-source library built on PyTorch[4] for NLP research. The models' hyperparameters are fine-tuned in the validation dataset.

4.4 Model Settings

We use 300 dimension GloVe (Pennington et al., 2014) as pre-trained word vectors and make it trainable during training. Adamax (Kingma and Ba, 2014) is our optimizer. To alleviate overfitting, we apply variational dropout (Srivastava et al., 2014) and clip the gradients (Pascanu et al., 2013) to the maximum norm of 5.0.

Additionally, Z_{init} is set as a vector of zeros, because the aim of this paper is to explore whether the reasoning mechanism can effectively utilize the relationships between labels, rather than the prior knowledge of the labels. More details please refer to the appendix.

5 Results and Analysis

5.1 ML-Reasoner v.s. Baselines Analysis

The experimental results of the ML-Reasoner and baseline models on the AAPD and RCV1-V2 test sets are shown in tables 3a and 3b respectively.

As shown in Table 3a, on the more challenging AAPD dataset, the ML-Reasoner outperforms all baselines by a large margin. Specifically, the hamming loss score of 0.0238 improves on the SGM+GE of 0.0245, a reduction of 2.9%; the micro-F_1 score is improved by an absolute margin of 1.0%.

Table 3b shows that our developed model also achieves state-of-the-art performance. Compared with SGM with global embeddigs, ML-Reasoner exhibits a 1.5% improvement over it in terms

[4]https://github.com/pytorch/pytorch

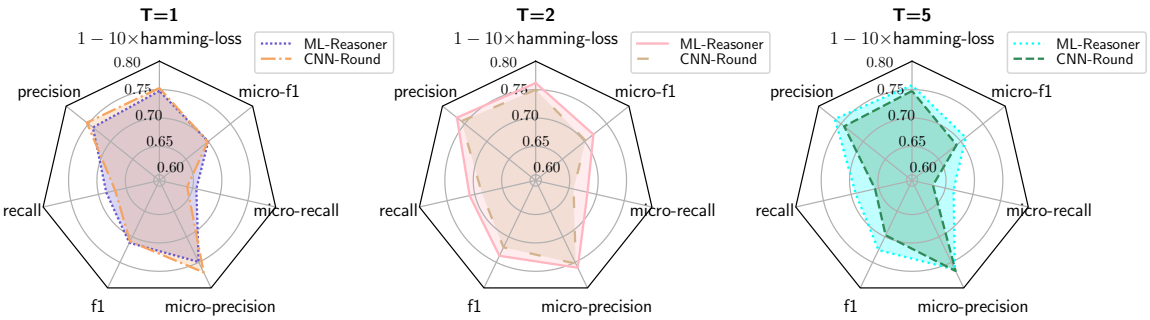

Figure 3: Performance radar maps on the AAPD test set for ML-Reasoner ($T = 1, 2, 5$) and CNN-Round ($T = 1, 2, 5$) respectively. Note that for the sake of appearance and the radar chart principle of *the larger the area, the better the performance*, the metric displayed on the graph is $1-10\times$hamming-loss instead of hamming-loss.

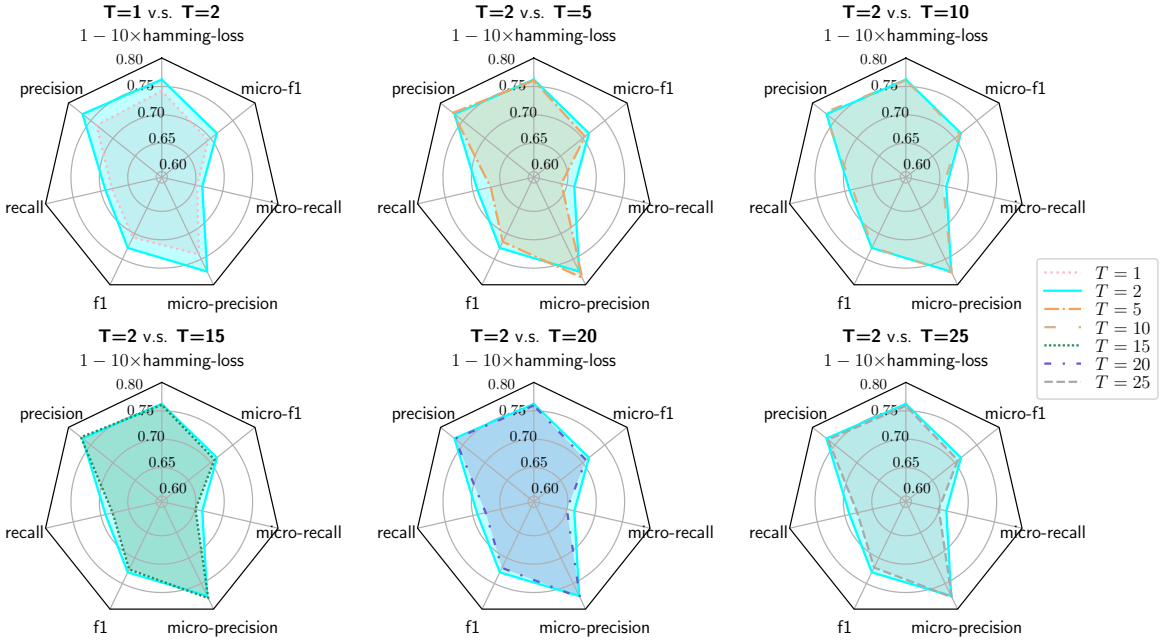

Figure 4: Comparison of performance on AAPD test set for ML-Reasoners with differing values for the number of rounds of reasoning. As before, $1-10\times$hamming-loss is used instead of hamming-loss.

of micro-precision, but has a lower micro-recall. This may be down to the RCV1-V2 dataset being easier to handle. Specifically, the label density of RCV1-V2 is 31.1% lower than that of AAPD while the number of its examples is about 14.5 times of that of AAPD.

5.2 Ablation Study

To explore whether the model's performance improvements are related to the use of the average loss rather than the reasoning mechanism, we compare ML-Reasoner with the internal iteration model removing reasoning mechanism named *CNN-Round*.

The CNN-Round uses the same text encoder and classifiers as the ML-Reasoner. It is also carried out T times for every round of predictions and optimized by the average loss in Eq. 12. The components *Label Encoder*, *Attention* and Combiner are removing from CNN-Round, because they all are related to the reasoning mechanism. The settings of both the CNN-Round model and the ML-Reasoner are entirely consistent.

The comparison between ML-Reasoners and CNN-Rounds with different T is shown in Fig. 3. As can be seen from the figure, because the ML-Reasoner with $T = 1$ actually does not make reasoning at all, all of the performance metrics for ML-Reasoner are very close to those of CNN-Round.

Fig. 3 also shows that increasing T does not significantly improve the performance of CNN-Round. But with T set to the same value for both models, ML-Reasoner outperforms CNN-Round

ID	True Labelset	before Reasoning	after Reasoning	Δ
1	math.IT, cs.IT	cs.IT	*math.IT*, cs.IT	↑
2	cmp-lg, cs.CL	cs.CL, *cs.FL*, cmp-lg	cmp-lg, cs.CL	↑
3	cs.SI, physics.soc-ph, cond-mat*	cs.SI, physics.soc-ph, *cs.CC*	cs.SI, physics.soc-ph, *cond-mat*	↑
4	cs.SC, cs.LO, cs.SY	*cs.LO*, cs.MS,cs.SC	cs.MS, cs.SC	↓
5	cs.SI, physics.soc-ph	cs.SI, *physics.soc-ph*, cs.IR	cs.SI, cs.IR	↓

Table 4: Some examples from the AAPD dataset showing the difference of performance as a result of reasoning. *True Labelset* is the set of actual labels for the example; *before Reasoning* is the prediction result before undergoing reasoning; *after Reasoning* is the prediction result after applying reasoning. The performance differences are represented by ↑, which indicates an increase in performance, and ↓, which indicates a decrease in performance. For brevity, we do not display the text data, which can be found in the appendix. * "cond-mat" stands for "cont-mat.stat-mech".

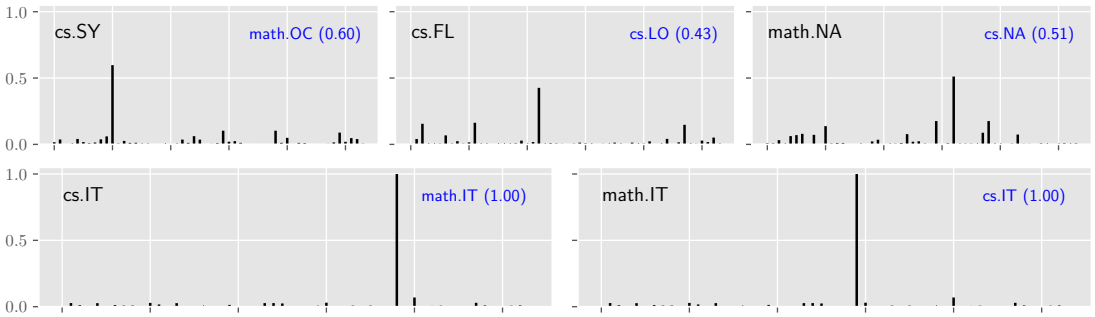

Figure 5: Frequency of co-occurring labels in the AAPD training set. For the purpose of appearance, we have left off the markings from the x-axis, that is the labels of the dataset. The values on the y-axis represent the frequency. The top right text shows the name and frequency of the label that has the most co-occurrence with each label.

by a large margin for most metrics.

The result reveals that the performance of MLC cannot be significantly improved if the loss is simply replaced by the average loss of multiple predictions. It also demonstrates the reasoning process in the ML-Reasoner is critical to the task of MLC.

5.3 The Impact of Reasoner Iteration Number T

Another question is the relation between the performance of ML-Reasoners and reasoner iteration number T.

For this purpose, we conducted experiments on the AAPD dataset with the ML-Reasoner T value set to $1, 2, 5, 10, 15, 20$ and 25. We conducted five experiments for each setting and then recorded the average performance, as shown in Fig. 4.

Fig. 4 shows that the area representing the performance for $T = 1$ is entirely covered by the area representing $T = 2$, while the areas for $T = 5, 10, 15, 20$ and 25 essentially coincide with each other.

Considering that a larger area covered in the radar chart indicates a better performance, this result illustrates that the ML-Reasoner with $T = 2$ outperforms one with $T = 1$ and further rounds of reasoning do not demonstrate an improvement in performance.

To sum up, what matters is *reasoning taking place*, not the number of rounds of reasoning.

5.4 Reasons For Reasoner Effectiveness

To further understand the role that reasoning plays in MLC tasks we have selected some examples from the AAPD dataset for which the predictions have changed as a result of reasoning, as shown in Table 4. In the first example, before reasoning takes place, the model predicts the label "cs.IT" to be relevant. Once reasoning has taken place, however, the model provides a more accurate and complete set of prediction results. In the second example, the model removes an irrelevant label from its set of predictions following the reasoning process. In the third example, the model both removes an irrelevant label from and adds a relevant label to its

prediction set following reasoning. In some cases reasoning also leads to inferior predictions, such as with examples 4 and 5 in Table 4.

In order to verify whether the changes shown in the examples above are due to consideration for the relationships between labels, we draw a graph displaying the frequency of label co-occurrence in the training set, as shown in Fig. 5. Fig. 5 shows that almost all labels depend on only one or a few labels. For example, in the training set, "cs.IT" and "math.IT" always appear in pairs. This helps explain how, in Example 1, the model is able to add the label "math.IT" based on the predicted "cs.IT". Additionally, "cs.FL" often appears with the label "cs.LO" in the training set (the co-occurrence frequency is 42.62%). In example 2, the label "cs.LO" was not included in the initial prediction results, which may be the reason why the model finally decided to delete "math.FL".

In order to quantitatively explore whether reasoning affects the behavior of the model in the ways described above, we collect the prediction results of the model on the training set and calculate the pre-reasoning and post-reasoning performance differences across various metrics, such as precision, recall and F_1.

Metric	$\Delta\uparrow$	$\Delta\downarrow$	**Mean**
F_1	40	14	$+0.0038$
precision	45	9	$+0.0137$
recall	39	15	-0.0073

Table 5: Summary of metric differences between the final predictions and the intermediate results without any reasoning. $\Delta\uparrow$ represents the number of label classes whose performance *improved*; $\Delta\downarrow$ represents the number of label classes whose performance *declined*. *Mean* refers to the average value in terms of difference in performance.

The summary of these differences is shown in Table 5. Table 5 shows that for almost all label classes, precision increased and for some label classes recall slightly decreased. These changes are consistent examples given in Table 4.

To summarize the above results, applying a reasoning mechanism *enables effective utilization of the relevant information between labels* in the task of MLC: adding relevant labels and removing irrelevant labels enables vast improvements in terms of the model's precision and performance.

6 Related Work

A number of algorithms have been proposed to handle MLC, which requires classifiers to predict multiple labels at once.

By binarization techniques, some methods (Gonçalves and Quaresma, 2003; Fürnkranz et al., 2008; Hüllermeier et al., 2008) decompose the MLC problem into multiple independent binary classification problems. However, it neglects the relationships between labels. In contrast, through reasoning, we are able to make use of the critical information among labels.

Instead of binary classifiers, LP-based approaches (Boutell et al., 2004; Read, 2008) use multi-class ones to model label correlations. Classifier-chains based methods (Read et al., 2011; Li and Zhou, 2013) link together single or multiple classifiers, where subsequent ones are built upon the predictions of preceding ones. Unfortunately, these methods either cannot handle unseen label combinations in training set, or ignore the fact that relationship between labels is not necessarily sequential. Unlike them, our method can handle all possible label combinations and avoid such an unreasonable method.

Recently, some neural network models are applied to MLC due to their effectiveness in classification tasks, such as Kurata et al. (2016); Chen et al. (2017). Yang et al. (2018) views MLC as a sequence generation problem to model the correlations between labels and achieves state-of-the-art performances on two datasets. But their method's assumption is likely to contradict the nature of MLC. As shown in 1, neither is the correlations between labels sequential nor mutually exclusive. Our method ML-Reasoner overcomes this drawback by a novel iterative reasoning mechanism when predicting.

7 Conclusions

In this paper, we introduce a reasoning-based algorithm for MLC, namely ML-Reasoner. It avoids generating an enlarged solution space and provides a new way to utilize label correlations without chaining classifiers. Extensive experimental results show that ML-Reasoners outperform competitive baselines. Additional analyses further reveal the contribution of iterative reasoning mechanism in ML-Reasoners and some empirical reasons for reasoner effectiveness.

References

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *IJCNN, 2017*, pages 2377–2383.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP, 2014*, pages 1724–1734.

Krzysztof Dembszynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2010. On label dependence in multilabel classification. In *ICML, 2010*.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Menc'ia, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153.

Mikel Galar, Alberto Fernández, Edurne Barrenechea Tartas, Humberto Bustince Sola, and Francisco Herrera. 2011. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44(8):1761–1776.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform. volume abs/1803.07640.

Teresa Gonçalves and Paulo Quaresma. 2003. A preliminary approach to the multilabel classification problem of portuguese juridical documents. In *EPIA, 2003*, pages 435–444.

Siddharth Gopal and Yiming Yang. 2010. Multilabel classification with meta-level features. In *SIGIR, 2010*, pages 315–322.

Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María José del Jesús. 2016. *Multilabel Classification - Problem Analysis, Metrics and Techniques*. Springer.

Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. 2008. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *PKDD, 2008*, volume 18.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *ECML, 2004*, pages 217–226.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *NAACL, 2016*, pages 521–526.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Nan Li and Zhi-Hua Zhou. 2013. Selective ensemble of classifier chains. In *MCS, 2013*, pages 146–156.

I. C. Mogotsi. 2010. Christopher d. manning, prabhakar raghavan, and hinrich schütze: Introduction to information retrieval - cambridge university press, cambridge, england, 2008, 482 pp, ISBN: 978-0-521-86571-5. *Inf. Retr.*, 13(2):192–195.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML, 2013*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP, 2014*, pages 1532–1543.

Jesse Read. 2008. A pruned problem transformation method for multi-label classification. In *Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, volume 143150.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359.

Robert E. Schapire and Yoram Singer. 1998. Improved boosting algorithms using confidence-rated predictions. In *COLT 1998*, pages 80–91.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Lena Tenenboim-Chekina, Lior Rokach, and Bracha Shapira. 2010. Identification of label dependencies for multi-label classification. In *Working Notes of the Second International Workshop on Learning from Multi-Label Data*, pages 53–60.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *PKDD, 2008*, volume 21, pages 53–59. sn.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. In *COLING, 2018*, pages 3915–3926.

Yiming Yang and Siddharth Gopal. 2012. Multilabel classification with meta-level features in a learning-to-rank framework. *Machine Learning*, 88(1-2):47–68.

Min-Ling Zhang and Kun Zhang. 2010. Multi-label learning by exploiting label dependency. In *KDD, 2010*, pages 999–1008.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Trans. Knowl. Data Eng.*, 18(10):1338–1351.

x

8 Supplemental Material

Module	Hyper-parameter	Value
Text Embedding	filter sizes	$3, 4, 5$
	number of each filter	all 100
	dimension of output vectors, D_1	600
CNN Encoder	filter sizes	$2, 3, 4, 5$
	number of each filter	all 100
	dimension of a sentence-level vector, D_2	400
	dimension of output vectors, D_3	300
Label Encoder	dimension of label embeddings, D_4	300
Classifiers	dimension of intermediate vectors, D_5	200
Trainer	number of epochs	300
	patience for early stopping	5
Adamax Optimizer	learning rate	$2\mathrm{e}{-3}$
	coefficients used for computing running averages of gradient and its square, βs	$0.9, 0.999$
	weight decay (L2 penalty)	0

Table 6: Details setting of ML-Reasoner. The dropout for the output of the text embedding and label encoder are set to a rate of 0.2, and the dropout for the first layer output of the MLP is set to a rate of 0.5.

ID	Text
1	methods for digital , phase coherent acoustic communication date to at least the work of stojanjovic , et al 20 , and the added robustness afforded by improved phase tracking and compensation of johnson , et al 21 this work explores the use of such methods for communications through tissue for potential biomedical applications , using the tremendous bandwidth available in commercial medical ultrasound transducers while long range ocean acoustic experiments have been at rates of under 100kbps , typically on the order of 1 10kbps , data rates in excess of 120mb s have been achieved over cm scale distances in ultrasonic testbeds 19 this paper describes experimental transmission of digital communication signals through samples of real pork tissue and beef liver , achieving data rates of 20 30mbps , demonstrating the possibility of real time video rate data transmission through tissue for inbody ultrasonic communications with implanted medical devices
2	a machine translation system is said to be complete if all expressions that are correct according to the source language grammar can be translated into the target language this paper addresses the completeness issue for compositional machine translation in general , and for compositional machine translation of context free grammars in particular conditions that guarantee translation completeness of context free grammars are presented
3	divorced individuals face complex situations when they have children with different ex partners , or even more , when their new partners have children of their own in such cases , and when kids spend every other weekend with each parent , a practical problem emerges is it possible to have such a custody arrangement that every couple has either all of the kids together or no kids at all ? we show that in general , it is not possible , but that the number of couples that do can be maximized the problem turns out to be equivalent to finding the ground state of a spin glass system , which is known to be equivalent to what is called a weighted max cut problem in graph theory , and hence it is np complete
4	one of the most frequently used models for understanding human navigation on the web is the markov chain model , where web pages are represented as states and hyperlinks as probabilities of navigating from one page to another predominantly , human navigation on the web has been thought to satisfy the memoryless markov property stating that the next page a user visits only depends on her current page and not on previously visited ones this idea has found its way in numerous applications such as google 's pagerank algorithm and others recently , new studies suggested that human navigation may better be modeled using higher order markov chain models , i e , the next page depends on a longer history of past clicks yet , this finding is preliminary and does not account for the higher complexity of higher order markov chain models which is why the memoryless model is still widely used in this work we thoroughly present a diverse array of advanced inference methods for determining the appropriate markov chain order we highlight strengths and weaknesses of each method and apply them for investigating memory and structure of human navigation on the web our experiments reveal that the complexity of higher order models grows faster than their utility , and thus we confirm that the memoryless model represents a quite practical model for human navigation on a page level however , when we expand our analysis to a topical level , where we abstract away from specific page transitions to transitions between topics , we find that the memoryless assumption is violated and specific regularities can be observed we report results from experiments with two types of navigational datasets (goal oriented vs free form) and observe interesting structural differences that make a strong argument for more contextual studies of human navigation in future work
5	we present abstraction techniques that transform a given non linear dynamical system into a linear system or an algebraic system described by polynomials of bounded degree , such that , invariant properties of the resulting abstraction can be used to infer invariants for the original system the abstraction techniques rely on a change of basis transformation that associates each state variable of the abstract system with a function involving the state variables of the original system we present conditions under which a given change of basis transformation for a non linear system can define an abstraction furthermore , the techniques developed here apply to continuous systems defined by ordinary differential equations (odes) , discrete systems defined by transition systems and hybrid systems that combine continuous as well as discrete subsystems the techniques presented here allow us to discover , given a non linear system , if a change of bases transformation involving degree bounded polynomials yielding an algebraic abstraction exists if so , our technique yields the resulting abstract system , as well this approach is further extended to search for a change of bases transformation that abstracts a given non linear system into a system of linear differential inclusions our techniques enable the use of analysis techniques for linear systems to infer invariants for non linear systems we present preliminary evidence of the practical feasibility of our ideas using a prototype implementation

Table 7: Texts of examples shown in Table 4.