Aggregating Intra-class and Inter-class Information for Multi-label Text Classification

Xianze Wu[1](✉), Dongyu Ru[1], Weinan Zhang[1](✉), Yong Yu[1], and Ziming Feng[2]

[1] Shanghai Jiao Tong University, Shanghai, China
{xzwu,maxru,wnzhang,yyu}@apex.sjtu.edu.cn
[2] China Merchants Bank Credit Card Center, Shanghai, China
zimingfzm@cmbchina.com

Abstract. This paper is concerned with the multi-label text classification (MLTC) task, whose goal is to assign one or more categorical labels to a document. The two critical characteristics of this task are the intra-class and inter-class information. The former means the distribution of samples belonging to the same category, and the latter models the relationships between labels such as label co-occurrence and label hierarchy. However, previous methods focus on either of them instead of combining both. This paper proposes a novel two-branch architecture to capture both intra-class and inter-class information. Experimental results show that considering both information improves the performance of the model. Besides, our model achieves competitive results on two widely used datasets.

1 Introduction

Multi-label text classification (MLTC) focuses on assigning one or multiple class labels to a document given the candidate label set. It has been applied to many fields such as tag recommendation [7], sentiment analysis [8], text tagging on social medias [18]. It differs from multi-class text classification, which aims to predict one of a few exclusive labels for a document [6].

Two types of information should be captured for the MLTC task. One is intra-class information, which cares the data distribution of samples belonging to the same category. The other is inter-class information, which models relationships between classes, such as label co-occurrence and hierarchy.

Prior efforts for multi-label text classification mainly focus on learning enhanced text representation [1,13,20,22]. These models feed the text representation into a set of linear classifiers. Each linear classifier predicts whether the given document belongs to a certain class. During training, the linear classifiers capture the intra-class information by learning the decision boundaries of corresponding classes. However, these methods neglect the inter-class information since the linear classifiers are trained independently and never interact with each other.

Recently, extracting the inter-class information has raised researchers' attention [15,16,19,23]. Some studies construct a label graph according to the inter-class information, and convert the graph into node features via random walk-based node embedding methods [23] or graph neural network (GNN) [15,16,19].

M. Tanveer et al. (Eds.): ICONIP 2022, CCIS 1791, pp. 239–250, 2023.
https://doi.org/10.1007/978-981-99-1639-9_20

The probability that a document belongs to a class is calculated by the dot product of document features and corresponding node features. These methods capture the inter-class information while depreciating the expressiveness of intra-class information. For node embedding-based methods, node embeddings are optimized in advance with the objective function of reconstructing neighbors. The optimized node embeddings, which take information for reconstruction rather than text classification, occupy the limited capacity originally used for modeling intra-class information. For GNN-based methods, the message passage process harms the expressiveness of intra-class information because the decision boundaries of classes receive noises from other nodes.

In this paper, we propose Aggregating Intra-class and Inter-class information Framework (AIIF) for MLTC. AIIF consists of a text encoder and a two-branch classification layer. On the classification layer, the linear branch applies multiple linear classifiers to capture intra-class information. The graph-assisted branch employs graph neural networks to a label-graph, where the message passing process captures the inter-class information. Each branch takes the text feature as input and makes predictions independently. Two branches' predictions are aggregated by a followed fusion module, which is optimized during the training process. With a divide-and-conquer architecture, AIIF captures both intra- and inter-class information and prevents the modeling of intra- and inter-class information from interfering with each other. Besides, AIIF supports plug-and-play usage, i.e., existing studies focusing on enhanced text representation or extracting inter-class information can be coupled with AIIF by serving as the text encoder or graph-assisted branch.

To evaluate the effectiveness of AIIF, we implement an instance of AIIF with BERT [5] and GCN [9], then evaluate the instance on widely used RCV1 and AAPD datasets. Experimental results show that the instance outperforms its variants without the two-branch classifier by a large margin. Besides, the instance achieves state-of-the-art results on the widely used RCV1 dataset and achieves competitive scores on the AAPD dataset.

The main contributions of this paper are listed as follows:

- We propose AIIF, a novel MLTC framework which can capture both intra-class and inter-class information.
- We implement an instance of AIIF. Experimental results show that the instance outperforms the baselines and gets competitive results on two public MLTC datasets.
- To our best knowledge, We firstly analyze MLTC from the view of intra- and inter-class information. We hope this work can provide a new perspective to the community.

2 Related Work

As mentioned in Sect. 1, existing MLTC work focuses on two directions: improving text representation and extracting the inter-class information.

To obtain a good text representation, many neural models have been applied, such as CNN [13], RNN [14,20,22], the combination of CNN and RNN [10], and BERT [1,3]. Some models consider improving text representation with the interaction between the input document and labels [6,20,22]. In addition, some methods construct a graph of words or documents to capture non-consecutive and long-distance semantics within a document or the whole corpus [17,21]. We argue that these methods neglect the inter-class information.

Capturing the inter-class information has attracted much attention in recent years. The main idea is modeling relationships between labels as graphs and guiding the multi-label prediction with graph representation. For example, Zhang et al. [23] construct a category graph according to the label correlations and used a random-walk-based method to encode the graph. Most subsequent work applies neural networks to encode label graphs, such as Tree-LSTM [24] and the variants of GCN [15,16,24]. Lu et al. [15] propose aggregating knowledge from multiple label graphs. Ma et al. [16] propose predicting the label graph according to each document. However, these methods sometimes perform worse than their variants without label graphs [2,15], which may be attributed to the reason that these methods ignore intra-class information.

Above methods only focus on either intra-class or inter-class information. Compared with them, AIIF applies a two-branch architecture to capture both information.

3 Methods

As shown in Fig. 1, AIIF separates the modeling of intra- and inter-class information with a two-branch classification layer. The classification layer takes the representation of the input document, which is obtained by the text encoder, as input. The linear branch captures intra-class information with a set of linear binary classifiers. The graph-assisted classifier branch models inter-class information by first encoding the label graph as node features, then calculating the dot product between node features and text features. The fusion module combines the predictions of the linear and graph-assisted branches as the probability that the input document belongs to each category.

Problem Formulation. Given the label space $\boldsymbol{\mathcal{L}} = \{\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_T\}$ and an input document $\boldsymbol{x}$, MLTC aims at predicting a label set $\boldsymbol{L_{pred}} \subseteq \boldsymbol{\mathcal{L}}$ for $\boldsymbol{x}$.

In the remaining part of this section, we first describe the working flow of AIIF when BERT serves as the text encoder and GCN serves as the graph encoder. Then we introduce the training of AIIF . Finally, we introduce the method of constructing the label graph.

3.1 AIIF Models

Text Encoder. Basically, given an input document $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i$ is the i-th token in the document, BERT converts $\boldsymbol{x}$ to the text feature as

$$\boldsymbol{H_t} = BERT(\boldsymbol{x}) \ , \tag{1}$$

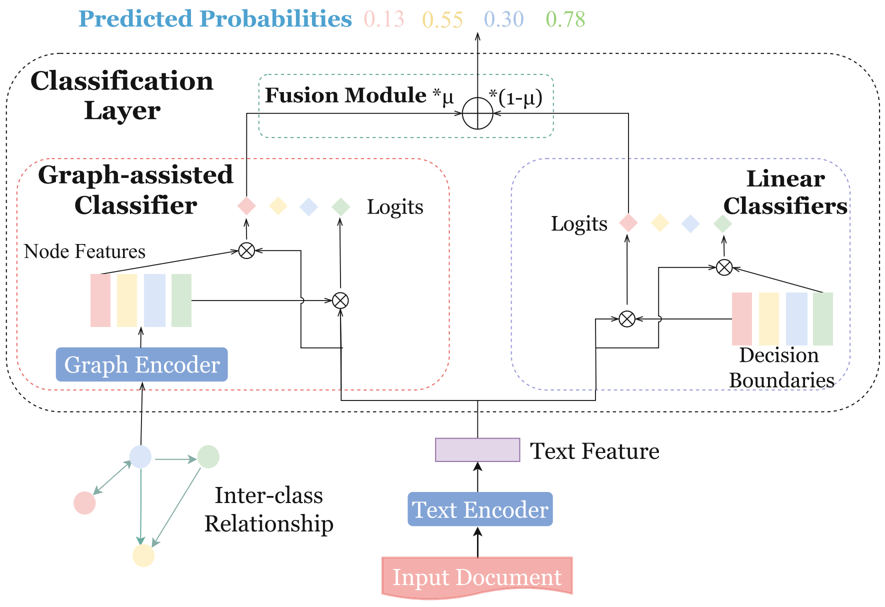

Fig. 1. The overall architecture of AIIF.

where $\boldsymbol{H_t} \in \mathcal{R}^{d_t}$ are the text feature. As recommended in [5], we add a special "[CLS]" token in front of $\boldsymbol{x}$ before feeding $\boldsymbol{x}$ into the BERT model, and take the token feature of the "[CLS]" token produced by the BERT model as the text feature of the input document $\boldsymbol{x}$.

Graph-assisted Classifier. Given the category graph $\mathbb{G}$, we obtain its node features via a graph convolutional network (GCN). We choose GCN for two reasons: First, GCN can capture the complex topology structure between labels through message passing. Second, GCN can be jointly trained with the other parts of the model in an end-to-end manner to achieve the optimal solution. We denote every convolutional network layer as a non-linear function $f(,)$, which takes the adjacency matrix $\boldsymbol{A'}$ and node features $\boldsymbol{H}_g^l \in \mathcal{R}^{T \times d_g^l}$ as input. Here, T is the number of nodes (labels) and d_g^l represents the dimension of node features.

$$\begin{aligned} \boldsymbol{H}_g^{l+1} &= f(\boldsymbol{H}_g^l, \boldsymbol{A'}) \\ &= \sigma(\boldsymbol{A'}\boldsymbol{H}_g^l\boldsymbol{W}^l) \end{aligned} \tag{2}$$

Here, $\boldsymbol{W}^l \in \mathcal{R}^{d_g \times d_g}$ is a weight matrix to be learned, $\sigma(\cdot)$ indicates a non-linear function, which is implemented with ReLU in this paper.

We treat the output of the last convolutional layer as node features, denoted by $\boldsymbol{L_G}$. The graph-assisted classifier produces the prediction for each label according to the dot product between text features and node features.

$$\begin{aligned} f^g(\boldsymbol{H_t}) &= \boldsymbol{H_t W^g} \\ z_i^g &= \boldsymbol{L_G}_i \cdot f^g(\boldsymbol{H_t})_i \end{aligned} \tag{3}$$

where $f^g()$ is a linear transformation with $\boldsymbol{W^g} \in \mathbb{R}^{d_t \times d_g}$. The transformation is required to match the dimension to that of L_G.

Linear Classifiers and Fusion Module. The linear classifiers make prediction for each label as,

$$\begin{aligned} f^p(\boldsymbol{H_t}) &= \boldsymbol{H_t W^p} \\ z_i^w &= f_p(\boldsymbol{H_t}) L_W \end{aligned} \tag{4}$$

where $f^p()$ is a linear transformation with $\boldsymbol{W^p} \in \mathbb{R}^{d_t \times d_p}$., L_W is the weight of linear classifiers.

The fusion module aggregates predictions of the linear classifiers z^w and that of the graph-assisted classifier z^g by weighted summation as follows,

$$z = z^g \cdot \mu + z^w \cdot (1 - \mu) \tag{5}$$

where $\mu \in \mathbb{R}^T$ is trainable parameters representing the ratio of inter-class relationship for each category.

3.2 AIIF Training

We adopt a 2-stage training for AIIF : (1) Training the text encoder (2) Training the classification layer.

Training the Text Encoder. In the first stage, we train the text encoder according to supervised signals from the dataset to obtain well-learned text features. Specifically, we add a linear classifier after the text encoder and compare predictions of the linear classifier with ground-truth labels with a hinge loss.

$$SH(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{T} (max(0, 1 - \boldsymbol{y}_i \cdot \hat{\boldsymbol{y}}_i)^2) \tag{6}$$

where $\hat{\boldsymbol{y}} \in \mathcal{R}^T$ is the prediction of the linear classifier. After training, only the parameters in the text encoder are saved for later use.

Training the Classification Layer. In the second stage, we freeze the parameters of the text encoder. The remaining parts of AIIF are randomly initialized and trained. Here, we use the binary cross-entropy (BCE) loss to train the model. Given the ground-truth label vector $\boldsymbol{y}$ and a vector of predicted probability p, the BCE loss is calculated as

$$BCE(p, \boldsymbol{y}) = \sum_{i=1}^{T} (\boldsymbol{y}_i \log p_i + (1 - \boldsymbol{y}_i) \log (1 - p_i)) \tag{7}$$

We apply BCE loss to the linear classifiers, the graph-assisted classifier and the whole model. The final loss L is calculated as

$$\begin{aligned} L_f &= BCE(sigmoid(z), \boldsymbol{y}) \\ L_w &= BCE(sigmoid(z_w), \boldsymbol{y}) \\ L_g &= BCE(sigmoid(z_g), \boldsymbol{y}) \\ L &= L_f + \alpha L_w + \beta L_g, \end{aligned} \tag{8}$$

where α and β are hyper-parameters used for balancing these losses.

For the initialization of the vertices embedding matrix, one method is using the mean-pooling of word embeddings of the tokens in the text of the label. However, some MLTC datasets does not provide label text or provide it in the form of abbreviation, which prevents us from obtaining a good initial vertices embedding matrix. Thus, we initialize the embedding of a category label according to the documents belong to the category. More specifically, if a set of documents $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$ have the ground-truth label $\boldsymbol{l}$, then the initial vertex embedding of $\boldsymbol{l}$ is

$$\boldsymbol{H_g^0} = \frac{1}{k} \sum_{j=1}^{k} BERT(x_j) \ , \tag{9}$$

3.3 Label Graph Construction

Following [4], we create the label graphs according to the co-occurrence patterns between labels within the dataset. Details are as follows.

First, we count the co-occurrence of label pairs $(\boldsymbol{l}_i, \boldsymbol{l}_j)$ in the training set to obtain the label correlation matrix $\boldsymbol{M} \in \mathbb{R}^{T \times T}$. Then, we calculate the conditional probability $P(\boldsymbol{l}_j|\boldsymbol{l}_i)$ that $\boldsymbol{l}_j$ appears when $\boldsymbol{l}_i$ appears as

$$P(\boldsymbol{l}_j|\boldsymbol{l}_i) = M_{ij}/N_i, \tag{10}$$

where N_i denotes the appearance frequency of $\boldsymbol{l}_i$ in the training set.

Then, we apply a threshold τ to filter out the noisy rare co-occurrence via

$$\boldsymbol{A}_{ij} = \begin{cases} 0, & \text{if } P(\boldsymbol{l}_j|\boldsymbol{l}_i) \ < \ \tau \\ 1, & \text{otherwise} \end{cases} , \tag{11}$$

where $\boldsymbol{A}$ is the binary adjacency matrix. However, directly applying $\boldsymbol{A}$ to GCN may cause the over-smoothing problem [12]. To alleviate the issue, we re-weight $\boldsymbol{A}$ as follows to obtain the final adjacency matrix $\boldsymbol{A'}$.

$$\boldsymbol{A'}_{ij} = \begin{cases} p/\sum_{j=1, i \neq j}^{C} A_{ij}, & if \ i \neq j, \\ 1-p, & if \ i = j \end{cases} \tag{12}$$

Table 1. Summary of the datasets. N is the number of samples in the training and validation set, M is the size of the testing set, W denotes the average length of documents, and $\tilde{C}$ means the average number of labels per sample.

Datasets	N	M	W	$\tilde{C}$
RCV1	23,149	781,265	223.2	3.2
AAPD	54,840	1,000	155.9	2.4

4 Experiments

4.1 Datasets and Evaluations

We perform experiments on two widely-used MLTC datasets: RCV1 [11] and AAPD. For a fair comparison, we follow the dataset split used in previous work [20]. The statistics of datasets are shown in Table 1.

Following the setting of previous work [16,20], we apply two metrics for performance evaluation: precision at top k **(P@k)**, the normalized discounted cumulated gain at top k **(nDCG@k)**. Given the ground-truth binary vector $\boldsymbol{y} \in \{0,1\}^T$, **P@k** is defined as follows:

$$P@k = \frac{1}{k}\sum_{l=1}^{k} \boldsymbol{y}_{rank(l)} \tag{13}$$

where $rank(l)$ is the index of the l-th highest predicted label. **nDCG@k** is defined as follows:

$$\begin{aligned} DCG@k &= \sum_{l=1}^{k} \frac{\boldsymbol{y}_{rank(l)}}{log(l+1)} \\ iDCG@k &= \sum_{l=1}^{min(k,||\boldsymbol{y}||_0)} \frac{1}{log(l+1)} \\ N@k &= \frac{DCG@k}{iDCG@k} \end{aligned} \tag{14}$$

4.2 Baselines

We select the following methods as baselines. **(1) DXML** Zhang et al. [23] construct a graph of labels considering the co-occurrence between labels and applied a random walk-based method to obtain node features. **(2) AttentionXML** You et al. [22] adopt a multi-label attention mechanism to perform hierarchical classification. **(3) LSAN** Xiao et al. [20] use the attention mechanism to consider the relations between document words and labels. **(4) LGAN** Ma et al. [16] use GCN to encode a static and dynamic text-specific label graph for predictions of each text. It achieved current state-of-the-arts results on RCV1 and AAPD datasets. We report the results of baselines from their original paper if no extra description.

4.3 Implementation Details

We apply BERT [5] as the text encoder because BERT has the ability to obtain strong contextualized text representation and has achieved great success in many NLP tasks. The pre-trained BERT utilized in this paper are provided by transformers[1] library. We use the BERT$_{base}$ checkpoint. We set $d_t = 768$ and $d_g = d_p = 256$. When constructing the label graph, we use $\tau = 0.5$ and $p = 0.2$.

For all training stages, we use the Adam optimizer. The initial learning rate lr is 5×10^{-4} expect for fine-tuning BERT in the first training stage, where lr is 5×10^{-5}. We use learning rate warm-up during the first 0.1 proportion of the whole training process, and a linear learning rate decay is applied for the remaining process. We use early stopping and the max training epoch for each stage is 20. We set $\alpha = 1$ and $\beta = 1$. Each document is truncated at the length of 350 and 250 for the RCV1 and the AAPD dataset, respectively.

Table 2. Compare AIIF with previous methods. † indicates that the scores are collected from [20]. Best results are shown in **bold**.

Dataset	RCV1					AAPD				
Method	P@1	P@3	P@5	nDCG@3	nDCG@5	P@1	P@3	P@5	nDCG@3	nDCG@5
DXML	94.04†	78.65†	54.38†	89.83†	90.21†	80.54†	56.30†	39.16†	77.23 †	80.99†
AttentionXML	96.41†	80.91†	56.38†	91.88†	92.70†	83.02†	58.72†	40.56†	78.01†	82.31†
LSAN	96.81	81.89	56.92	92.83	93.43	85.28	61.12	41.84	80.84	84.78
LDGN	97.12	82.26	57.29	93.80	95.03	86.24	61.95	**42.29**	**83.32**	**86.85**
BERT (ours)	97.18	83.36	57.61	94.19	94.48	86.60	62.40	41.58	81.75	85.20
AIIF	**97.50**	**84.07**	**58.22**	**94.85**	**95.20**	**86.9**	**62.40**	42.06	82.34	85.81

4.4 Main Results

We compare AIIF with previous methods. Results are shown in Table 2. From the results, we can observe that

1. **AIIF significantly outperforms BERT.** AIIF outperforms BERT in all metrics in both datasets, with the improvement between 0.70% and 1.15% in RCV1 dataset and between 0.23% and 1.44% in AAPD dataset. The superiority of AIIF over BERT is that AIIF capture both intra- and inter-class information, not just intra-label information.
2. **Competitive results on two datasets.** Compared with the previous state-of-the-art method LDGN, AIIF outperforms LDGN in all metrics on the RCV1-v2 dataset, with an improvement between 0.27% and 2.35%; on the AAPD dataset, AIIF achieved close results with LDGN, with an improvement between −1.20% and 0.73% in each metric.

[1] https://github.com/huggingface/transformers.

4.5 Ablation Study

Table 3. AIIF compares with its variants on MLTC datasets. *AIIF-L* represents the model consists of the text encoder and linear classifiers. *AIIF-G* represents the model consists of the text encoder and graph-assisted classifier. The best results are shown in **bold**.

Dataset	RCV1					AAPD				
Methods	P@1	P@3	P@5	nDCG@3	nDCG@5	P@1	P@3	P@5	nDCG@3	nDCG@5
AIIF	**97.50**	**84.07**	**58.22**	**94.85**	**95.20**	**86.80**	62.40	**42.06**	**82.34**	**85.81**
AIIF-L	97.34	83.65	57.67	94.48	94.63	86.10	**62.50**	41.84	82.13	85.52
AIIF-G	96.40	83.99	58.17	94.53	94.89	85.80	62.37	41.90	82.09	85.51

To further analyze the effectiveness of the two-branch architecture, we compare AIIF to its two variants: (1) **AIIF-L**: We remove the graph-assisted classifier and the fusion module from the AIIF. The linear classifiers' predictions are treated as the final prediction. (2) **AIIF-G**. We remove the linear classifiers and the fusion module from the AIIF. The graph-assisted classifier's predictions are treated as the final prediction. AIIF, AIIF-L, and AIIF-G follow the same training methods.

The results are shown in Table 3. We can observe that in most cases, removing any branch of AIIF will cause the performance drop on two datasets. Take the P@1 score on the AAPD dataset as an example, AIIF-L is inferior to AIIF by 0.81%, and AIIF-G is inferior to AIIF by 1.27%. The performance drop demonstrates the effectiveness of the proposed two-branch classifier.

4.6 Performance on the Tail Labels

As mentioned in Sect. 1, previous work shows that encoding the inter-class information achieves promising results on tail labels [19]. We are interested in whether introducing the intra-class information further improves results on tail labels. Thus, we evaluate AIIF and its variants with propensity scored precision at k (PSP@k), which is calculated as

$$PSP@k = \frac{1}{k}\sum_{l=1}^{k}\frac{y_{rank}(l)}{P_{rank}(l)} \tag{15}$$

Results are shown in Fig. 2. We can observe that AIIF outperforms AIIF-G on all datasets, demonstrating that even if a model has captured the inter-class information, introducing the intra-class information still improves performance on tail labels.

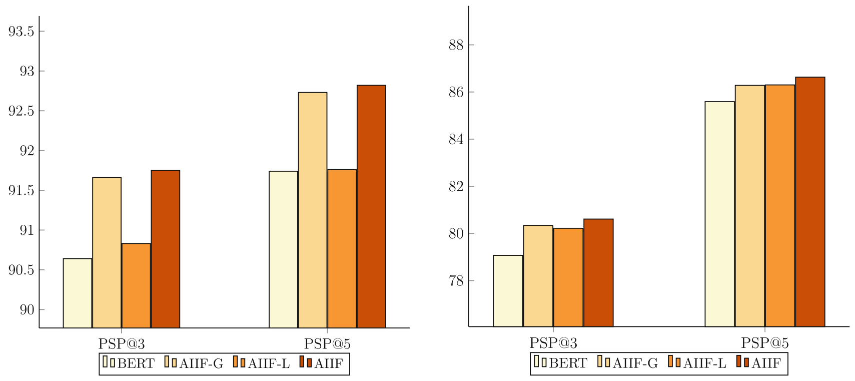

Fig. 2. Performance on tail labels

Table 4. A case study on the RCV1 dataset. Here, we show the top 5 predictions of BERT and AIIF, and the right predictions are colored by red.

Input Document: French pension at 55 ups cost 117 bln
Reducing the french retirement age to 55 from 60 could cost as much as 117 billion francs over 15 to 20 years, an unpublished study by the pensions branch of the social security system says, according to Daily Les Echos. The newspaper said a preliminary study by the CNAV branch of the social security system believed retirement across the board at 55 would add 28 million people to the pensioner population, raising existing claimers 31 percent from 91 million. The CNAV estimated that the extra burden on the basic social security retirement benefit system in extra payouts would be in the region of 100 billion francs.,
...
Ground-truth labels: economics; government/social; expenditure/revenue; government finance; welfare, social services
Top 5 predictions of BERT: government/social; health; domestic politics; expenditure/revenue; welfare, social services
Top 5 predictions of AIIF: government/social; welfare, social services; expenditure/revenue; economics; government finance

4.7 Case Study

Further, we compare the prediction results of AIIF and BERT to analyze why AIIF is superior to BERT in Table 4.

For the example shown in Table 4, BERT correctly predicts the categories "government/social," "expenditure/revenue," and "welfare, social services," but not the categories "economics" and "government finance". AIIF predicts all the correct categories. We believe the phenomenon can be attributed to AIIF extracting the relationship between the categories after introducing the category map. In the training set, the categories "economics" and "government finance" have a strong co-occurrence with the three categories correctly predicted by BERT. Correspondingly, they are connected by edges in the category graph. Extracting such inter-class information increases the probability of predicting “economics” and “government finance”. Conversely, the two categories that BERT incorrectly predicted, “health” and “domestic politics”, have weaker co-occurrence with the three categories that BERT correctly predicted. Hence, AIIF excludes these two incorrect predictions.

5 Conclusion

This paper studies the multi-label text classification task. Previous methods focus either intra-class or inter-class information. We propose a novel two-branch architecture to combine both information. Experimental results show that the model capture both intra-class and inter-class outperforms those modeling either of them.

References

1. Adhikari, A., Ram, A., Tang, R., Lin, J.: Docbert: bert for document classification. arXiv preprint arXiv:1904.08398 (2019)
2. Chalkidis, I., Fergadiotis, M., Kotitsas, S., Malakasiotis, P., Aletras, N., Androutsopoulos, I.: An empirical study on large-scale multi-label text classification including few and zero-shot labels. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, 16–20 November 2020, pp. 7503–7515. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.emnlp-main.607
3. Chang, W.C., Yu, H.F., Zhong, K., Yang, Y., Dhillon, I.: X-bert: extreme multi-label text classification using bidirectional encoder representations from transformers. In: Proceedings of NeurIPS Science Meets Engineering of Deep Learning Workshop (2019)
4. Chen, Z.M., Wei, X.S., Wang, P., Guo, Y.: Multi-label image recognition with graph convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5177–5186 (2019)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Du, C., Chen, Z., Feng, F., Zhu, L., Gan, T., Nie, L.: Explicit interaction model towards text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6359–6366 (2019)
7. Fürnkranz, J., Hüllermeier, E., Mencía, E.L., Brinker, K.: Multilabel classification via calibrated label ranking. Mach. Learn. **73**(2), 133–153 (2008)
8. Gopal, S., Yang, Y.: Multilabel classification with meta-level features. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 315–322 (2010)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

10. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: Twenty-ninth AAAI Conference on Artificial Intelligence (2015)
11. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: a new benchmark collection for text categorization research. J. Mach. Learn. Res. **5**, 361–397 (2004)
12. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
13. Liu, J., Chang, W.C., Wu, Y., Yang, Y.: Deep learning for extreme multi-label text classification. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 115–124 (2017)
14. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101 (2016)
15. Lu, J., Du, L., Liu, M., Dipnall, J.: Multi-label few/zero-shot learning with knowledge aggregated from multiple label graphs. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, 16–20 November 2020, pp. 2935–2943. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.emnlp-main.235
16. Ma, Q., Yuan, C., Zhou, W., Hu, S.: Label-specific dual graph neural network for multi-label text classification. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 3855–3864 (2021)
17. Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., Yang, Q.: Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 World Wide Web Conference, pp. 1063–1072 (2018)
18. Ren, Z., Peetz, M.H., Liang, S., Van Dolen, W., De Rijke, M.: Hierarchical multi-label classification of social text streams. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 213–222 (2014)
19. Rios, A., Kavuluru, R.: Few-shot and zero-shot multi-label learning for structured label spaces. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (eds.) Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pp. 3132–3142. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/d18-1352
20. Xiao, L., Huang, X., Chen, B., Jing, L.: Label-specific document representation for multi-label text classification. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 466–475 (2019)
21. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 7370–7377 (2019)
22. You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H., Zhu, S.: Attentionxml: label tree-based attention-aware deep model for high-performance extreme multi-label text classification. arXiv preprint arXiv:1811.01727 (2018)
23. Zhang, W., Yan, J., Wang, X., Zha, H.: Deep extreme multi-label learning. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, pp. 100–107 (2018)
24. Zhou, J., et al.: Hierarchy-aware global model for hierarchical text classification. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 1106–1117 (2020)