Legal Knowledge and Information Systems
G. Sileno et al. (Eds.)

doi:10.3233/FAIA230966

New Horizons of Legal Judgement Predication via Multi-Task Learning and LoRA

REN-DER SUN [a], CHIA-HUI CHANG [b,1], and KUO-CHUN CHIEN [c]
[a] *renkensun40@gmail.com*
[b] *chiahui@g.ncu.edu.tw*
[c] *qk0614@gmail.com*
ORCiD ID: Ren-Der Sun https://orcid.org/0009-0008-2351-5353, Chia-Hui Chang https://orcid.org/0000-0002-1101-6337, Kuo-Chun Chien https://orcid.org/0009-0000-0441-0363

Abstract.
Legal Judgment Prediction (LJP) aims to predict the judgement results (such as legal article, charge and penalty) based on the criminal facts of the case. Most previous research in this field was based on criminal statements from court verdicts. However, each verdict actually is based on the content from indictments. For prosecutors, will the case be dismissed or processed? If the case is accepted, is the penalty a jail sentence or a fine? What is the charge and article violated? In this study, we therefore define three novel LJP tasks for prosecutors, including prosecution outcome prediction (LJP#1), imprison prediction (LJP#2) and fine prediction (LJP#3). We explore various multi-task learning (MTL) framework based on Word2Vec and BERT language model (LM) with either topology-based or message-passing mechanism. Moreover, we employed the LoRA (Low-Rank Adaptation) technique to save both computation time and resources during fine-tuning. Experimental results demonstrated that Word2Vec-based model combined with message passing architecture still has the potential to outperform large LM like BERT, while BERT-based models with a simple parallel architecture generally performed well. Finally, using LoRA for fine-tuning not only reduced training time (by 45%) but also improved performance (2.5% F1) in some LJP tasks.

Keywords. Legal Judgement Prediction, Large Language Model, LoRA, PEFT

1. Introduction

Legal artificial intelligence (AI) has a wide range of applications, including legal judgment prediction, similar case matching, legal question answering, etc. In this study, we focus on the task of Legal Judgment Prediction (LJP). A typical LJP task involves predicting the legal articles, charge, and term of penalty simultaneously based on given criminal fact, thus it often require a multi-task learning (MTL) framework. In the past,

[1]Corresponding Author: Chia-Hui Chang, Dept. of Computer Science & Information Engineering, National Central University, Taiwan. Email: chia@csie.ncu.edu.tw

most related studies use the fact description recorded in the **court verdicts** as input (e.g. the CAIL dataset [1]In practice, both **indictments** and **verdicts** of criminal cases are open data from Ministry of Justice Procuratorate Agency Public Document Query System. Although using the criminal facts in the judgment as input may more accurately predict the outcome of the case, the judgments could not be used as input in real situation. Rather, we are more concerned with the consequences of prosecution. In this study, our main goal is to predict possible outcomes following prosecution, which can assist young prosecutors in anticipating potential judgment results. For example, will the case be sustained or dismissed? If the case is dismissed, what are the grounds for its rejection? If sustained, what judgment the defendant might face, will the punishment involves imprisonment or fined? The first three questions formulate our first LJP task, prosecution outcome prediction (LJP#1). Furthermore, in criminal sentencing, judges can impose fines instead of imprisonment for less serious cases. Therefore, the second and third LJP tasks deal with imprisonment cases (LJP#2) and fine (LJP#3) penalties respectively. Similar to the CAIL dataset, the two LJP tasks include three subtasks: legal article, charge, and term prediction. Since Taiwan's criminal law consists of general laws and special laws, such as the Securities Exchange Law and the Law on the Administration of Firearms, Ammunition and Knives, we introduced the subtask of law prediction. The definitions of the three LJP tasks are illustrated in Figure 1, where the numbers under each subtask indicate the number of labels in each subtask. Besides, predicting charges has a very important purpose, because the "charge" charged by the prosecutor during the prosecution stage may be different from the "charge" decided by the judge during the sentencing stage.

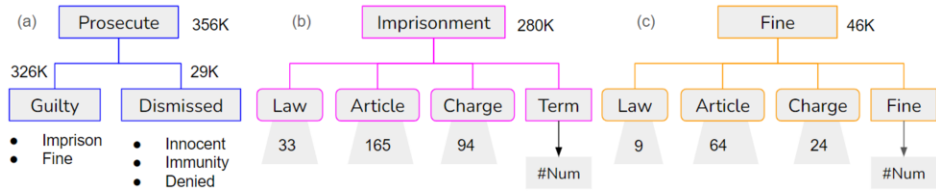

Figure 1. Three LJP Tasks (The numbers indicate the number of labels for each subtask.)

Since LJP is essentially a multi-task learning (MTL) framework and there are topological dependencies between these legal subtasks, Zhong et al. [2] proposed a topological multi-task learning framework named TopJudge to improve judgment prediction by incorporating dependencies through directed acyclic graph (DAG). On the other hand, He et al. [3] introduced iterative message passing networks (IMN) to pass information through shared latent variables between subtasks of sentiment analysis to make better predictions. However, both TopJudge and IMN were proposed based on shallow (small) language models such as Word2Vec. It is unclear whether these topology-based structures or message-passing networks are still useful for the transformer-based language model BERT. Meanwhile, since we have three LJP tasks, if we train a large language model separately for each LJP task by fully fine-tuning all parameters, the cost is triple in terms of time, memory and storage. To save the cost, we introduce the LoRA (Low-Ranked Adaptation) [4] architecture, a technique in PEFT (Parameter Efficient Fine-tuning) [5], which can reduce trainable parameters without sacrificing performance, thus save computing resources and reducing training time, or even gain better performance

than full tuning. To sum up, we study the Multi-Task Learning (MTL) framework and try several MTL structures such as TopJudge, IMN, and their combinations, along with various language models including Word2Vec and BERT. We analyze the performance of different topologies and language model combination to find the optimal configuration for three LJP tasks defined in this paper. Additionally, we apply LoRA for fine-tuning to the prediction of above tasks. The experimental results show that the Word2Vec-based model and appropriate topology architecture still have the potential to outperform large-scale language models (in LJP1), but BERT-based models usually only require the simplest parallel architecture model to perform well (LJP2 and 3). Finally, fine-tuning with LoRA not only reduces training time (by 45%) but also improves performance on a specific LJP task (2.5% F1). The contributions of this paper are as follows:

- We define three novel LJP tasks from prosecutors perspective to address practical scenarios encountered by prosecutors in their work, including prosecution result prediction, fined prediction, and imprisonment prediction.
- We explore various multi-task learning architectures with different language models. By leveraging the dependencies between different subtasks, we transfer relevant knowledge to dependent subtasks.
- We exploit the advantage of LoRA to reduce the training time and storage space requirements caused by large language models.

2. Related Work

Multi-task learning is a machine learning paradigm where a model is trained to perform multiple related tasks simultaneously, rather than training separate models for each individual task. The idea is that jointly learning multiple tasks can improve the model's overall performance compared to training separate models. This is achieved by sharing and leveraging common features or representations across tasks, which can help tasks reinforce each other and lead to better generalization. In classic MTL, all tasks are parallel and independent. However, when there are dependencies between tasks, the prediction results of other tasks may help predictions for other tasks. Therefore, Zhong et al. [2] proposed a MTL framework called TopJudge, which uses inter-task dependencies to help predict the results of other sub-tasks within the model. The dependencies between subtasks are connected with nodes to form a directed acyclic graph (DAG). The information flows to other subtasks along the connection established by the directed graph, realizing information sharing between subtasks and assisting the model to make legal judgment predictions. In addition to MTL, Xu et al. [6] proposed an graph neural network called GDL and an attention mechanism to extract discriminative features from fact descriptions to overcome the difficulties of distinguish confusing articles. On the other hand, Gan et al. [7] injected the legal knowledge as a set of first-order logic rules and integrate these rules into a co-attention network-based model, which enhances model with explicit logical reason capabilities and makes the prediction result even more interpretable.

3. Problem Definition of three LJP task

Given $D = \{F_1, F_2, \cdots, F_n\}$ as a dataset containing n cases from indictments, where each case F_i is composed of a series of m words, $F_i = (w_1^i, w_2^i, \cdots, w_m^i)$. Depending

on the judgement outcome from court verdicts, a case is associated with three labels: $s_i \in \{Sustained, Dismissed\}$, $r_i \in \{Innocent, Immunity, Denied, Sustained\}$, and $p_i \in \{Imprisonment, Fine, Dismissed\}$. If a case is rejected, then p_i is labeled with *Dismissed*. If a case is sustained, then r_i is labeled with *Sustained*. The case description F_i and the three labels form the LJP#1 task. For LJP#2 task, all cases with $p_i = Imprison$ will also be processed to obtain the associated law $l_i \in R^p$, articles $a_i \in R^r$, charge $c_i \in R^q$ and prison term $t_i \in R$, where p and q denote the size of the one-hot vector for law and charge, while a_i is a multi-hot vector of articless with dimension r. Finally, LJP#3 has similar labels as that of LJP#2 but a fine value instead of prison sentence value.

4. Method

In this paper, we introduce an alternative approach based on message passing to exploit subtask dependency, and low rank adaptation (LoRA) for large language models to reduce the number of parameters to be fine-tuned. As with other MTL models, all categorical subtasks (including all subtasks in LJP#1 as well as the law, articles, and charge prediction task) are optimized with cross entropy loss function as shown in Eq.(1), where M denotes the number of classes, $y_{o,c}$ is a binary indicator (0 or 1) representing whether class label c is the correct classification for observation o, and $p_{o,c}$ is the predicted probability that observation o is of class c.

$$\text{Cross Entropy Loss} = \sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

As for imprisonment and fine prediction tasks, log square loss function is used as shown in Eq.(2), where y_o is the true value of the example o and $\hat{y}_o$ is the predicted value.

$$\text{Log Square Loss} = (\log(y_o + 1) - \log(\hat{y}_o + 1))^2 \tag{2}$$

4.1. Message-Passing Model

Message passing is a commonly used mechanism in graph neural networks. It has also been applied in interactive multi-task networks (IMN)[3] for aspect-based sentiment analysis. As shown in Figure 2, the feature extraction component encode input fact descriptions into embedding vectors, which is then fed to each task-specific component for prediction. The output results of each task are returned and combined to the shared latent vector through the message passing mechanism. After completing the update of the shared latent vector, the next round of iterations starts until the given number of iterations is completed. In addition, subtasks with interdependencies, the dependence of articles on law, the dependence of charges on articles, the dependence of penalty on articles, will be linked by attention mechanism.

4.2. IMN+TopJudge

The IMN model is specifically designed to address the issue of insufficient interaction between independent subtasks in multi-task learning. It employs a Message-Passing mecha-

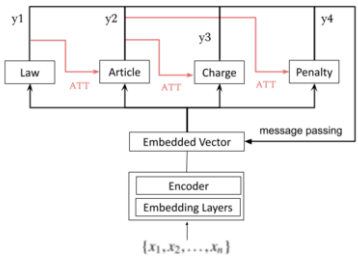

Figure 2. Message Passing Architecture

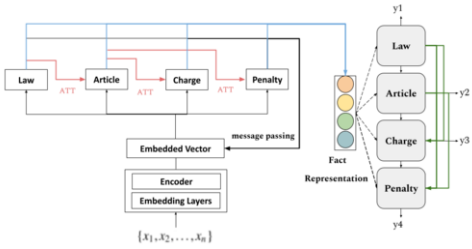

Figure 3. IMN+TopJudge Architecture

nism to enhance interaction by allowing the shared embedded vector to propagate among subtasks. This interaction is critical to improving the performance of the LJP model in gathering information from different subtasks. However, the mutual dependencies among subtasks cannot be fully enforced even if we incorporate sentencing logic into the model through attention mechanisms. Therefore, we integrate IMN and TopJudge to ensure the final judgement predictions. As shown in the figure, the fact representation output of the IMN module is input into the TopJudge module, which uses the topological relationship between subtasks to enhance predictions. It is hoped that this integration, in addition to utilizing sub-task interactions, will also be consistent with the final sentencing logic, ensuring the prediction accuracy and interpretability of the LJP task.

4.3. Low Rank Adaptation for Large Language Model

Please note that both TopJudge and IMN are models proposed before BERT, and their input is encoded using shallow word embedding model, Word2Vec. Therefore, the model parameters are fewer and is difficult to capture rich contextual information and semantic nuances. On the other hand, BERT [8] is a deep learning model based on the Transformer architecture. The bidirectional attention mechanism and deep architecture enable it to capture complex linguistic relationships and dependencies, while also producing models with hundreds of millions or even billions of parameters. Although the pre-

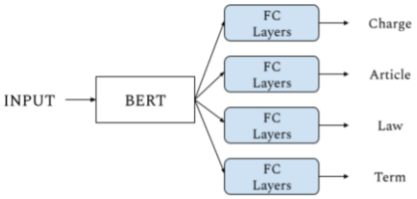

Figure 4. BERT+Parallel Architecture

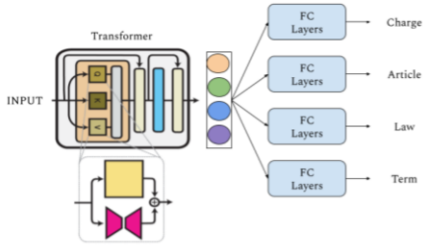

Figure 5. LoRA Architecture

train and fine-tune paradigm of large language models has been widely accepted, fine-tuning all parameters usually requires a large amount of computing resources. In order to overcome the challenges in fine-tuning LLM, we introduce Parameter-Efficient Fine-

Tuning[5] technique, which freezes the pre-trained language model (PLM) by selectively updating only a subset of parameters and enabling you to get performance comparable to full fine-tuning. In this paper, we specifically utilize Low-Rank Adaptation (LoRA) to enhance PLM by converting attention mechanisms into low-rank representations, resulting in fewer trainable parameters. As shown in Figure 5, by incorporating LoRA modules within the Transformer architecture, the input data is transformed into embeddings and passed to subtasks, and connected through Fully-Connected Layers to offer a more efficient approach to Fine-Tuning models.

5. Experiment

We collected criminal prosecution cases and their corresponding first-instance judgments from the Judicial Yuan's judgment system in Taiwan, spanning from June 15, 2018, to June 30, 2021 through web scraping. Fortunately, most legal text structures have regularity, we observe the text structure and use regular expressions to extract the charges, articles, law recorded in the indictment and combined with the penalty recorded in the corresponding first-instance judgment. This information was then stored in JSON format. The code that we extract these labeled can be viewed in the following link [2]. In addition, since the interpretation of the law may change over time, we use cases before 2021 as training data and cases after 2021 as testing data. It's important to note that in practice, a single case might violate multiple article, but prosecutors typically list the most severe charge in the indictment. Therefore, while we have more article than charge names, predicting charges is more challenging than predicting article.

Task	#Cases	#Law	#Articles	#Charges	#Train	#Valid	#Test
TWLJP#1	356,588	-	-	-	285,183	35,640	35,765
TWLJP#2	280,402	33	165	94	224,326	28,039	28,087
TWLJP#3	46,292	9	64	24	37,024	4,627	4,641

Table 1. TWLJP Dataset

Table 1 describes the datasets we prepared for the three LJP tasks. TWLJP#1 is used to predict prosecution results, including three subtasks: Task1 - whether there is punishment (2 labels), Task2 - reasons for not punishing (3+1 labels), Task3 - punishment type (2+1 labels) . TWLJP#2 contains all cases with prison terms, including the violated laws (33 tags), articles (164 tags), charges (94), and prison terms. Finally, TWLJP#3 focuses on cases where only fines are imposed as penalties. The subtasks in TWLJP#3 resemble those in TWLJP#2, except for the fine prediction task. Besides, it is worth noting that the number of legal articles (64), charges (24) is comparatively smaller in TWLJP#3. For the evaluation of law, article, charge, or subtasks in TWLJP#1, we use Macro F1 as shown in Eq.(3) as the evaluation metric. As for the prediction of fines and term of imprisonment, we employ Log-Distance[9] as shown in Eq.(4) as the evaluation metric, where O_i is the output value and L_i is the Label value.

[2] https://drive.google.com/file/d/12BHkJKTuQ_JJDrKvTIuM9aKJEUKQhr09/view?usp=sharing

$$\text{Macro F1} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i}\right) \tag{3}$$

$$\text{Log Distance} = \Sigma_{i=1}^{n}\left(\frac{|\log(O_i+1) - \log(L_i+1)|}{n}\right) \tag{4}$$

In the following experiment, we conduct LJP#1 and LJP#2 for 10 epoch and LJP#3 for 50 training epoch. While using BERT as Language model, we set the hidden size with 768 and maximum length with 512. The optimizer is BERT Adam with learning rate 1e-5; When using LoRA for fine-tuning the BERT model, the value of rank is set to 8. And we choose AdamW as the optimizer with the learning rate of 3e-4, maximum length of 512 and hidden size of 768 for the parameters of pre-trained language model. While using Word2Vec as the Language model, the hidden size is 768 but the maximum length change to 1024 and we use Adam as the optimizer with the learning of 1e-4. In each experiment, we selected the epoch with the best performance on the validation dataset and measured the performance on the testing dataset. The performance shown in the tables is the average performance of the model over three experiments with its standard deviation.

5.1. Experiment Result of LJP#1

Table 2 presents the experimental results of two language models combined with different topology on LJP#1 dataset. It is surprising to see that small (53M) traditional models based on Word2Vec with appropriate topology architecture design could outperform large-scale language models. One possible explanation is that the task of LJP#1 is simple, so large language models do not take advantage. As for fine-tuning the BERT-based models using low-rank adaptation, the training time is significantly reduced (39 to 49%) in all cases, as shown in Table3. In terms of prediction performance, LoRA can improve the performance of most subtasks, but sometimes sacrifices the performance of other subtasks.

TWLJP1-Prosecution Result Prediction-Apply different topology						
LM	Topology	#Para	Time	Task1(Punish)	Task2(Reason)	Task3(Penalty)
				MacroF1	MacroF1	MacroF1
Word2Vec	TopJudge	140M	45mins	**98.96±0.05**	**95.66±0.75**	**71.53±0.41**
	IMN	91M	66mins	98.53±0.11	94.66±0.40	73.90±0.17
	IMN+TopJudge	141M	46mins	98.63±0.25	71.96±0.35	66.93±0.64
BERT	TopJudge	153M	198mins	95.03±0.37	**87.36±0.15**	**69.96±0.87**
	IMN	103M	197mins	94.80±0.10	64.83±0.40	65.20±1.24
	IMN+TopJudge	154M	205mins	**98.76±0.15**	67.20±0.62	63.30±0.17

Table 2. TWLJP#1-Prosecution Result Prediction-Apply different topology

5.2. Experiment Result of LJP#2

For the more challenging task of LJP#2, the performance of IMN+TopJudge architecture based on Word2Vec performs the best, surpassing the individual performance of IMN

TWLJP#1-Prosecution Result Prediction-Apply LoRA or not						
Topology/Embedding	LoRA	#Para	Time	Task1	Task2	Task3
				MacroF1	MacroF1	MacroF1
Parallel/BERT	N/A	102M	192 mins	93.13±1.75	64.26±1.43	51.03±4.74
	+LoRA	0.3M	98 mins	**95.20±0.17**	**78.13±0.8**	**74.33±1.62**
TopJudge/BERT	N/A	153M	198 mins	95.03±0.37	**87.36±0.15**	69.96±0.87
	+LoRA	51M	110 mins	**95.13±0.20**	85.90±1.05	**70.76±0.47**
IMN/BERT	N/A	103M	197 mins	94.80±0.10	64.83±0.40	65.20±1.24
	+LoRA	1.5M	115 mins	**95.20±0.17**	**76.56±1.30**	**69.13±0.11**
IMN+TopJudge/BERT	N/A	155M	205 mins	**98.76±0.15**	67.20±0.62	63.30±0.17
	+LoRA	52M	125 mins	95.43±0.11	**87.53±0.15**	**70.33±0.23**

Table 3. TWLJP#1-Prosecution Result Prediction-Apply LoRA or not

and TopJudge (see Table 4). As for the BERT-based models, a simple TopJudge architecture outperforms the others, highlighting that LLM-based models do not necessitate overly complex structures. Overall, the IMN+TopJudge architecture based on Word2vec also outperforms TopJudge based on Bert (except for penalty prediction). On the other hand, as shown in Table 5, the performance improvement of LoRA for simple architectures is not as good as that for complex architectures. In other words, for simple architectures such as Parallel and TopJudge architectures, models without LoRA perform better than models with LoRa. For complex architectures, such as IMN and IMN+TopJudge, models using LoRA perform better than models without LoRA. Furthermore, when using BERT-based models, the simplest parallel architecture outperforms other models, just like LJP#1.

TWLJP2-Imprisonment Prediction-Apply different topology							
LM	Topology	#Para	Time	Law	Article	Charge	Term
				MacroF1	MacroF1	MacroF1	LD
Word2Vec	TopJudge	140M	56mins	89.05±0.21	66.00±0.56	64.75±0.49	0.76±0.01
	IMN	91M	90mins	89.90±0.65	69.90±0.36	63.90±0.26	0.75±0.01
	IMN+TopJudge	142M	142mins	**93.46±0.20**	**74.53±0.41**	**67.16±0.40**	**0.75±0.00**
BERT	TopJudge	153M	207mins	**93.33±0.87**	**64.80±0.98**	**65.33±1.40**	**0.72±0.01**
	IMN	103M	205mins	81.20±0.36	50.96±0.32	54.53±0.37	0.76±0.01
	IMN+TopJudge	154M	221 mins	89.03±0.23	49.10±0.36	56.40±0.36	0.75±0.01

Table 4. TWLJP#2-Imprisonment Prediction-Apply different topology

5.3. Experiment Result of LJP#3

Finally, the LJP#3 task has the smallest training data set (37K instances) among the three problems. Therefore, we train the model for 50 epochs. Overall, when using Word2Vec as the language model, all topologies perform better than their corresponding topologies based on BERT (see Table 6). However, when using BERT as the LM, a simple parallel structure can achieve better performance than complex topologies such as IMN, TopJudge, IMN+TopJudge. In addition, freezing BERT-based LM and adding LoRA still have comparable performance / or will not drop much in performance, but it will reduce a lot of training time and computing resources (as shown in Table 7).

TWLJP#2-Imprisonment Prediction-Apply LoRA or not							
Topology/Embedding	LoRA	#Para	Time	Law	Article	Charge	Term
				MacroF1	MacroF1	MacroF1	LD
Parallel/BERT	N/A	102M	201 mins	**93.33±0.25**	**73.20±0.40**	68.06±0.45	0.74±0.01
	+LoRA	0.7M	115 mins	93.16±0.20	70.70±0.17	**69.80±0.30**	**0.72±0.01**
TopJudge/BERT	N/A	153M	207 mins	**93.33±0.87**	64.80±0.98	**65.33±1.40**	**0.72±0.01**
	+LoRA	51M	116 mins	90.63±0.41	**66.20±0.55**	62.03±0.66	0.77±0.01
IMN/BERT	N/A	104M	205 mins	81.20±0.36	50.96±0.32	54.53±0.37	0.76±0.01
	+LoRA	2.8M	120 mins	**92.96±1.04**	**63.30±0.98**	**63.30±0.98**	**0.75±0.01**
IMN+TopJudge/BERT	N/A	154M	221 mins	89.03±0.23	49.10±0.36	56.40±0.36	0.75±0.01
	+LoRA	52M	135 mins	**89.93±0.37**	**64.60±1.24**	**59.66±0.25**	**0.74±0.01**

Table 5. TWLJP#2-Imprisonment Prediction-Apply LoRA or not

TWLJP3-Fined Prediction-Apply different topology							
LM	Topology	#Para	Time	Law	Article	Charge	Term
				MacroF1	MacroF1	MacroF1	LD
Word2Vec	TopJudge	141M	10mins	**96.67±0.40**	**76.07±2.15**	**72.43±1.04**	**8.32±0.02**
	IMN	91M	15mins	95.03±1.72	68.83±3.37	68.67±1.36	8.44±0.02
	IMN+TopJudge	143M	20mins	**96.87±0.38**	**74.13±2.19**	**71.33±0.68**	8.30±0.01
BERT	TopJudge	153M	28mins	96.87±0.61	68.73±4.24	69.67±2.25	**8.21±0.04**
	IMN	104M	27mins	90.33±4.80	63.30±3.36	**67.30±1.91**	**8.28±0.05**
	IMN+TopJudge	155M	27mins	**95.77±2.42**	**64.00±2.33**	66.57±1.97	8.31±0.10

Table 6. TWLJP#3-Fined Prediction-Apply different topology

TWLJP#3-Fined Prediction-Apply LoRA or not							
Topology/Embedding	LoRA	#Para	Time	Law	Article	Charge	Money
				MacroF1	MacroF1	MacroF1	LD
Parallel/BERT	N/A	103M	27 mins	**97.73±0.51**	**71.93±1.89**	**73.50±0.35**	9.11±0.03
	+LoRA	0.4M	16 mins	96.57±0.65	71.77±3.34	71.07±0.55	**8.43±0.03**
TopJudge/BERT	N/A	153M	28 mins	**96.87±0.61**	**68.73±4.24**	**69.67±2.25**	**8.21±0.04**
	+LoRA	51M	16 mins	95.37±0.99	64.43±4.63	67.40±0.79	8.34±0.02
IMN/BERT	N/A	104M	27 min	90.33±4.80	63.00±3.36	**67.30±1.91**	**8.28±0.05**
	+LoRA	1.9M	15 mins	**95.03±0.50**	**64.17±2.44**	65.87±2.83	8.32±0.03
IMN+TopJudge/BERT	N/A	155M	29 mins	**95.77±2.42**	**64.00±2.33**	**66.57±1.97**	**8.31±0.10**
	+LoRA	53M	16 mins	95.17±1.16	62.33±2.51	61.60±1.23	8.34±0.04

Table 7. TWLJP#3-Fined Prediction-Apply LoRA or not

6. Conclusion and Future Work

In this paper, we define three novel tasks from a prosecutor's perspective and use indictments as input to address difficulties that prosecutors may encounter in their work. We explore different topologies and various language models for multi-task learning. Experimental results show that for simple multi-task learning like LJP#1 or when the training data set is large enough like LJP#2, Word2Vec-based model when combined with appropriate topology can outperform BERT-based models. For BERT-based models, simple parallel structures often outperform other complex architectures. In addition, Bert-based models also outperform Word2Vec-based models when the amount of training data is limited. Finally, applying LoRA can reduce the number of trainable parameters by approximately 24% and save nearly 50% of training time. For future work, we find that the

performance of current models sometimes sacrifices the performance of some subtasks to achieve the performance of other subtasks. How to ensure that each sub-task has a certain effectiveness is the direction of our future efforts.

References

[1] Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and Jianfeng Xu. Cail2018: A large-scale legal dataset for judgment prediction, 2018.

[2] Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3540–3549, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. . URL https://aclanthology.org/D18-1390.

[3] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 504–515, Florence, Italy, July 2019. Association for Computational Linguistics. . URL https://aclanthology.org/P19-1048.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[5] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[6] Nuo Xu, Pinghui Wang, Long Chen, Li Pan, Xiaoyan Wang, and Junzhou Zhao. Distinguish confusing law articles for legal judgment prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3086–3095, Online, July 2020. Association for Computational Linguistics. . URL https://aclanthology.org/2020.acl-main.280.

[7] Leilei Gan, Kun Kuang, Yi Yang, and Fei Wu. Judgment prediction via injecting legal knowledge into neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12866–12874, May 2021. . URL https://ojs.aaai.org/index.php/AAAI/article/view/17522.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. . URL https://aclanthology.org/N19-1423.

[9] Junyun Cui, Xiaoyu Shen, Feiping Nie, Zheng Wang, Jinglong Wang, and Yulong Chen. A survey on legal judgment prediction: Datasets, metrics, models and challenges, 2022.