Dependency Learning for Legal Judgment Prediction with a Unified Text-to-Text Transformer

Yunyun Huang*, Xiaoyu Shen*, Chuanyi Li, Jidong Ge, Bin Luo

Abstract—Given the fact of a case, Legal Judgment Prediction (LJP) involves a series of sub-tasks such as predicting violated law articles, charges and term of penalty. We propose leveraging a unified text-to-text Transformer for LJP, where the dependencies among sub-tasks can be naturally established within the auto-regressive decoder. Compared with previous works, it has three advantages: (1) it fits in the pretraining pattern of masked language models, and thereby can benefit from the semantic prompts of each sub-task rather than treating them as atomic labels, (2) it utilizes a single unified architecture, enabling full parameter sharing across all sub-tasks, and (3) it can incorporate both classification and generative sub-tasks. We show that this unified transformer, albeit pretrained on general-domain text, outperforms pretrained models tailored specifically for the legal domain. Through an extensive set of experiments, we find that the best order to capture dependencies is different from human intuitions, and the most reasonable logical order for humans can be sub-optimal for the model. We further include two more auxiliary tasks: court view generation and article content prediction, showing they can not only improve the prediction accuracy, but also provide interpretable explanations for model outputs even when an error is made. With the best configuration, our model outperforms both previous SOTA and a single-tasked version of the unified transformer by a large margin. Code and dataset are available at https://github.com/oli-yun/Dependency-LJP.

Index Terms—NLP in Law, Legal Judgment Prediction, Dependency Learning, Neural Networks.

1 Introduction

Legal Judgment Prediction (LJP) aims to predict the judgment results of legal cases according to the fact descriptions, which involves multiple sub-tasks depending on country-specific standards. Under the Civil Law system, these sub-tasks usually include (1) finding the violated law articles, (2) defining the charge, and (3) deciding the term of penalty. Automating these sub-tasks is of great interest in that it can not only improve the working efficiency of judges and lawyers, but also provide basic legal assistance to non-professionals.

Earlier works solved these sub-tasks as independent text classification problems, while ignoring the close dependencies among different sub-tasks [1]–[5]. Recently, many works have demonstrated benefits of modelling such dependencies. The simplest way is a pipeline method that conditions each sub-task on the prediction of its dependent sub-tasks [6]–[9]. However, it requires an independent sub-module for each sub-task, which complicates the overall system and prevents information sharing across sub-tasks. Another line of work is multi-tasking by sharing the same model parameters among all sub-tasks [10], [11]. Nonetheless, the prediction of each sub-task is still based solely on the fact description. The dependency learning is only implicitly reflected through parameter sharing while ignoring the logical orders of sub-tasks. A few works enable both parameter sharing and dependency learning under specified logical orders [12]–[14]. They design different architectures for each purpose, e.g., CNN for fact encoding and LSTM for dependency learning, such that all sub-tasks can share the same fact encoder while maintaining their own task-specific representations.

We argue that this way of combining different architectures is by no means the optimal solution, and instead propose leveraging a unified text-to-text Transformer for dependency learning in LJP. Specifically, we implement our model based on the pretrained T5 [15] architecture, where all sub-tasks are considered as “masked spans” from the original legal judgment document. In the training stage, the model learns all sub-tasks by reconstructing “masked spans” sequentially. The generated sequence can be easily mapped to classification labels for the evaluation purpose. Compared with previous works, it has the following advantages: (1) It fits in the same pattern as the pretraining objective of the T5 model. The structure of the legal judgment documents can naturally serve as semantic prompts [16], [17] for different sub-tasks, which provides the model with a much better initialization especially under the fewshot scenario. (2) There will be no separate architectures or task-specific representations. It utilizes a single unified Transformer, enabling full parameter sharing across all sub-tasks. (3) It applies to both classification and generative tasks, so that there will be no limitation on the sub-task types.

Experiments on real-world criminal cases show that our proposed unified architecture, albeit pretrained on general-domain text, outperforms pretrained models tailored specif-
applied for the legal domain. It performs especially well on the Macro-F1 metric, implying a strong generalization on low-frequency labels. In contrast, all previous works overfit to high-frequency labels without exception. It also demonstrates strong few-shot learning capability with only hundreds of training examples. We conducted an extensive set of experiments on the effects of sub-task orders under dependency learning. We find that the best order is different from human intuitions, and the most reasonable logical order for humans can be sub-optimal for the model. The model is seeking a sweet spot to balance well the dependency on other sub-tasks and error propagation. When the risk of error propagation outweighs the benefit from dependency learning, it prefers putting ahead the sub-task, even if it should be solved in a later logical order for humans.

Apart from the three commonly used sub-tasks, we introduce two more auxiliary tasks: court view generation and law article content prediction. For the former, we show it is able to not only improve the prediction accuracy, but also provide interpretable explanations to the model. It is able to not only improve the prediction accuracy, but also provide interpretable explanations to justify the logic behind, even when the model makes a wrong prediction. For the latter, we find that it boosts the prediction accuracy when combining with every single task. In the law article prediction sub-task, it even outperforms the text-matching model which learns a full interaction between facts and law article contents, while being hundreds of times faster.

In short, this work makes the following contributions:

- We propose leveraging a unified text-to-text Transformer for dependency learning in LJP. We show it significantly outperforms previous SOTA with a much stronger generalization on low-frequency labels, and performs decently even under the few-shot scenario.
- We perform an extensive set of experiments to find the optimal order for dependency modelling. We find the best order for the model depends on the trade-off between sub-task dependency and error propagation, and can be different from human intuitions.
- We introduce two new auxiliary tasks: court view generation and law article content prediction. They can further enhance the prediction accuracy, while providing interpretable explanations to the model outputs.

2 Problem Formulation

We focus on LJP tasks under the civil law system, but similar techniques can be easily extended to other systems as well. Given factual descriptions of one case, LJP under the civil law system involves a set of sub-tasks, including predicting the violated law articles, charges and term of penalty. The violated law article is legal basis for judgment results and charge is the category of the committed crime, e.g., theft, intentional injury, etc. We illustrate the process of LJP in Figure 1. For human judges, there exist dependencies with a strict order among these sub-tasks. A judge will always first decide the violated law articles, then determine the charges according to the instructions of relevant law articles. Finally, the term of penalty will be confirmed based on these.

The violated law article and charge have a pre-defined fixed set of possible labels, but the term of penalty can be rather diverse across different crimes. Following common practices, we simplify it into three main types: fixed-term imprisonment, life imprisonment and death penalty. For fixed-term imprisonment, the model needs to predict the exact length in the unit of month. Since it is stipulated that the maximum period of fixed-term imprisonment is 15 years and that of the criminal who commits several crimes is 25 years, we set the value of life imprisonment as 350 and death penalty as 400 in accordance with [12].

Formally speaking, we denote our dataset with $n$ cases as $D = (F^i, \{A^i_1, \ldots, A^i_k\}, \{C^i_1, \ldots, C^i_l\}, P^i)_{i=1}^n$ where each case $D^i$ consists of a fact description $F^i$, a set of $k$ violated law articles $\{A^i_1, \ldots, A^i_k\}$, a set of $l$ accused charges $\{C^i_1, \ldots, C^i_l\}$ and a penalty term $P^i$. All of these items are composed of a sequence of words. Law articles and charges have their corresponding sets of categories and each law article has its own textual definition. Our goal is to learn a classifier $\zeta$ which is able to predict the judgment results for a given fact, i.e., $\{\{A^i_1, \ldots, A^i_k\}, \{C^i_1, \ldots, C^i_l\}\} = \zeta(F^i)$.

3 Methods

In this section, we first introduce the T5 model and semantic prompts, then describe how we can leverage the existing legal judgment documents to construct semantic prompts for LJP based on T5. Finally, we explain how it enables easy dependency learning for arbitrary sub-tasks.

3.1 Introduction of T5

We construct our model based on T5 [15], a large-scaled pretrained language model (PLM) based on the Transformer architecture [18]. At each time step $t$, it uses a left-to-right language model $D$ to predict $y_t$, conditioned on previous generated tokens and hidden states $h$ from a separate self-attention-based encoder $E$ for input $x$:

$$h = E(x) \quad y_t = D(h, y_1, \ldots, y_{t-1})$$

T5 unifies the training framework of natural language understanding and natural language generation into the text-to-text form. It is pretrained with the span-corruption objective, where consecutive spans of input tokens are replaced with a mask token and the model is trained to reconstruct the masked-out tokens. During pretraining, the input is the masked sentence and the target is a sequence of...
3.2 Semantic prompts

Applying semantic prompts to PLMs has become increasingly popular. Under this paradigm, downstream tasks are reformulated to look more like those solved during the original PLM pretraining objective with the help of a textual prompt. For example, when recognizing the emotion of a social media post, “I missed the bus today.”, we may continue with a prompt “I felt so ______”, and ask the LM to fill the blank with an emotion-bearing word [19]. The filled word can be mapped to emotion labels by pre-defined rules. It has shown consistent improvement over the traditional supervised finetuning especially in the fewshot scenario [16], [20] because it can map tasks into existing semantic knowledge inside the PLM instead of learning from scratch.

3.3 Prompts For LJP

It is quite straightforward to construct prompts for LJP. Since all sub-task labels are themselves extracted from legal judgment documents by regex rules [21], we can directly leverage the structure of the document as prompts for different sub-tasks. A legal judgment document starts with a fact, following by a set of judgment results and explanations. We can mask out the words describing the judgment results from the document the same regex rule as used in [21], input this corrupted document into the T5 model, then ask the model to predict original spans in order. An illustration is shown in Figure 2. By this means, the training shares exactly the same format as the original pretraining objective of T5, such that we can make the most use of the knowledge inside the pretrained model. After the model predicts the sequence, we can convert it into sub-task labels with a one-to-one mapping.

3.4 Dependency Learning

The dependency learning across sub-tasks is naturally established with the autoregressive decoder from the T5 model, where the prediction of one span conditions on all the previously decoded spans. Taking predicting law articles and term of penalty as example, we suppose that the latter depends on the former here. We first use a prompting function to modify the input fact $F_i$ into a prompt:

$$\tilde{F}_i = f_{\text{prompt}}(F_i)$$

and the output is the concatenation of different sub-tasks labels which are connected by their indicators:

$$Y^i = \{I_1; A_{i1}; \ldots; A_{ik}; I_2; P_i; I_3\}$$

where $I_k$ represents the indicator of the $k$-th sub-task and the last one is the end signal. Then we pass $\tilde{F}_i$ into T5 and get output tokens step by step. When the model decode $I_2$ at time step $t$, we can use previous generated tokens $\{y_1, \ldots, y_{t-1}\}$ to map corresponding law articles $\{A_{11}, \ldots, A_{ik}\}$, and these tokens are also used to predict term of penalty which is equivalent to:

$$P^i = D(\mathcal{E}(\tilde{F}_i), A_{11}, \ldots, A_{ik})$$

In this way, we build dependencies between different sub-tasks. Same as in text generation tasks, the decoder conditions on the ground-truth spans during training, and on the self-predicted spans during inference. The order of spans in the decoding side reflects the dependency relations. We can simply adjust the order of decoded spans (together with its indicator) to control the logical order of sub-tasks.

3.5 Auxiliary Tasks

The above framework provides us a unified way to incorporate arbitrary sub-tasks existing in the legal judgment document. We can easily add on more tasks by masking out the corresponding spans from the original document. In this work, we experiment with two more auxiliary tasks: court view generation and law article content explanation. Court
Fig. 3. Three types of learning methods used in LJP. Single-task learning applies independent models for different tasks. Multi-task learning applies a single model for all tasks, but there is no dependency across tasks. Dependency-learning applies a single model to capture all dependencies within tasks.

view can be regarded as the interpretation for the sentence of a case. It summarizes the fact description and specifies related charges, then explains the rationale to derive the judgment. Law article content includes a complete description of the premises for violation and the scope of sentence. Both of them can be useful for LJP since they provide the background basis and explanations on how the judgment results are derived. By co-training with these two auxiliary tasks, the model is able to "explain" its own predictions by generating the corresponding court view and article content, which is crucial for a trustworthy LJP system.

4 EXPERIMENTS

We conduct our experiments on a dataset collected from the China Judgments Online. In this section, we first explain how we collect such dataset, then go over a set of baseline systems we compare with, and finally present the findings.

4.1 Dataset

In the field of LJP, existing works often experimented on the CAIL dataset or extracted task-specific texts from the published judgment documents to construct their own dataset. However, none of them contains all the 5 tasks we need. Furthermore, these datasets have been through a series of pre-processing which makes it difficult to map each case to the original legal judgment document. Therefore, we construct a dataset ourselves following the same mechanism as in [21]. First of all, we filtered those cases involving multiple defendants because different defendants correspond to different trial results. Besides, the top 102 law articles in Chinese Criminal Law are not relevant to specific charges, we also filtered these labels. In this way, we got a dataset involving 200 charges and 183 law articles from 225,843 criminal judgment documents. We randomly selected 12,810 cases to form the test set and ensured it covered all types of charges and law articles. We also selected 12,634 cases to construct our validation set and the rest is training set. Each case refers to 1.14 law articles and involves 1.06 charges on average, but has strictly one term of penalty. We provide more detail statistics in Table 1.

|      | Fact | Article | Charge | Penalty | View | Content |
|------|------|---------|--------|---------|------|---------|
| # Unique | 225,843 | 183 | 200 | 207 | 225,843 | 183 |
| # Words  | 199.46 | 3   | 7.26 | 6.41 | 148.28 | 137.9 |

4.2 Experimental Settings

4.2.1 Model configuration

We fine-tune our model based on mT5 base, a multilingual variant of T5 that can be used to fine-tune on Chinese tasks. For training, we set batch size as 128 and adopt the gradient accumulation strategy. All models are trained for a maximum of 30 epochs with early-stopping, and the model which performs best on the validation set will be selected. Max length of input and output are both set to 512. We use the AdaFactor optimizer which can relieve memory pressure to a certain extent.

4.2.2 Metrics

For law article and charge, we employ micro-F1 and macro-F1 as evaluation metrics. Since penalty is regarded as regression task, we calculate the log-distance between prediction and ground-truth following [25], a smaller log-distance indicates better prediction.

4.2.3 Baselines

We compare with the following baselines:

- **LSTM** [26]: We employ two-layer bidirectional LSTM to encode factual description and then make prediction for each task with full-connected neural network.
- **CNN** [27]: We employ CNN with multiple filter widths as fact encoder and then make prediction.
- **FactLaw** [6]: It predicts top-k law articles with SVM, then encodes fact and law article contents with HAN and predicts the result.
- **TopJudge** [12]: It encodes facts with CNN, and then models task dependencies with LSTM.
- **LBERT** [29]: It pretrained BERT on a large scale number of Chinese criminal judgment documents which shows better performance than original BERT in LJP. We use this model to encode fact following by

1. https://wenshu.court.gov.cn/
three differently connected networks for prediction of main tasks respectively.

FactLaw did not release the code, so we set \( k = 5 \) and reproduce it ourselves. Other baselines are implemented with the open-sourced code in [25]. For all above models, we compare them under the following settings:

- **Single-task Learning (STL)**: As shown in Figure [3a], it applies independent models for different tasks. There is no correlation among tasks.
- **Multi-task Learning (MTL)**: As shown in Figure [3b], it trains all tasks with a single model with parameter sharing, but there is no dependency across tasks.
- **Dependency Learning**: As shown in Figure [3c], it trains all tasks with parameter sharing plus explicit dependency modelling. Only FactLaw and TopJudge can apply to this setting.

### 4.3 Main Results

We show the main comparison results in Table 2. All models are trained on the same 10,000 subset sampled from the whole training data as a pilot study. For the dependency-learning models, we use the order of article-charge-penalty for model prediction, which, as explained, is the order that humans follow to make the judgment results. For the experiments with auxiliary tasks, we append the auxiliary tasks in the end of the three main tasks.

As for the model architecture, pretrained LBERT and T5 have a clear advantage compared with other models that are trained from scratch. T5 also significantly outperforms LBERT, even though it has never been tailored for legal-specific text. T5 performs especially well on the Macro-F1 metric, with a consistent lead of \( 8 \sim 30\% \), suggesting it is able to generalize to low-frequency labels effectively, while all other models overfit only to high-frequency labels. For all architectures, MTL does not lead to significant improvement over STL on most tasks, suggesting parameter sharing only is not an effective way for dependency learning.

When applying dependency learning, both the performance of TopJudge and T5 get improved. However, the improvement of Dependent-T5 over STL/MTL-T5 is much larger than that of Dependent-TopJudge over STL/MTL-TopJudge, indicating our proposed structure is a more efficient way to enable dependency learning than TopJudge.

By adding the two auxiliary tasks individually, the performance of Dependency-T5 is again improved. However, combining both of them did not help further. The benefit that could get from auxiliary tasks could have been saturated.

### 4.4 Analysis

#### 4.4.1 Orders and Dependencies

We show the effects of decoding orders in Figure [4] where we modify the orders of three main tasks and insert two auxiliary tasks into different positions. As can be observed, dependency learning benefits not only successor tasks, but also predecessor tasks. Law article prediction under all the 6 order combinations outperform the single-task performance, suggesting the unified dependency learning framework provides a more effective way of parameter sharing than multi-task learning. Even for the first decoding task that has no dependency on other tasks, it can still outperform multi-task learning under the same setting.

The prediction of article and charge correlate more with each other, and knowing one of them helps significantly the prediction of the other. Penalty can benefit more from article/charge than the other way around. This coincides with humans that the decision of penalty should be left at the end.

The best decoding order for each task can be different from human intuitions. The reason is that there exists error propagation for machines. When the downside of error propagation outweighs the dependency of other tasks, machines will prefer putting ahead the task to maximize the performance. For example, humans tend to decide the penalty after confirming the charge, yet the model prefers predicting...
Fig. 4. Effects of decoding orders. For visualization, we divided the term of penalty into 11 categories to calculate Macro-F1 too. Bars in different colors indicate different orders of decoding article ($A$), charge ($C$), penalty ($P$), court view ($V$) and article content ($A'$). Upper bars in lighter colors are results of making prediction based on the ground-truths of predecessor tasks. Dashed lines indicate single-task results.

![Graph showing different decoding orders of main tasks, different slot positions of court view, and different slot positions of law article content.]

The penalty before the charge because the error propagation from charges will negatively impact the prediction.

This is more obvious for auxiliary tasks. For example, humans will first come with the court view because it includes the derivation that we need to decide the penalty. For the model, however, putting the court view before the penalty significantly deteriorates the results because the court view is long and prone to error propagation. The same holds for law article contents. For both auxiliary tasks, the best order is to put them at the end so that they will not incur any error propagation for the main tasks.

### 4.4.2 Law Article Content

The law article item and content have a strictly one-to-one relation. A more common way to leverage the context for article prediction is by text matching, where we turn it from multi-label classification to multiple binary classification problems. Following the common practice for text matching, we concatenate the fact with every single law article content as the input of T5, and decode a binary relevant/irrelevant output. We take the positive-negative ratio as 1:31 to construct training set and train it with cross-entropy loss following [31]. When testing, the classification threshold is set as 0.7 (tuned to maximized the micro-F1). We compare it with the STL-T5 (predict article only) and Dependent-T5 (decode article then content). Results are shown in Table 3. We can see that Dependent-T5 outperforms the text-matching method without affecting the inference speed because the content prediction is only used in training. Text matching, on the contrary, slows down the inference significantly as it needs to match over all articles one by one. Even with parallel batching, it still doubles the time with hundreds more memory cost.

### 4.4.3 Data Size

To observe the effect of data size on the dependency learning advantage, we trained TopJudge, STL-T5, and dependent-T5 (with the identified best decoding order $A$, $C$, $P$, $V$) on training sets of varying sizes. The results on test set are shown in Figure 5. With the increase of training data, performances of three models continue to improve. When training data grows, the gap between STL-T5 and Dependent-T5 becomes smaller. We conjecture that the effects of parameter sharing and dependency learning will saturate with sufficient data. Nonetheless, both outperform TopJudge by a large margin.

![Graph showing performance by training on varying data sizes.]

**TABLE 3**

Comparison of leveraging law article contents for article prediction (all models are based on mT5). Dependent-T5 is more effective than text matching while being significantly faster.

| Methods                     | MiF  | MaF  | Speed/s |
|-----------------------------|------|------|---------|
| STL (article)               | 0.757| 0.429| 0.37    |
| Text Matching               | 0.770| 0.431| 64.74   |
| Dependent (article + content)| 0.817| 0.488| 0.37    |
Fig. 6. Visualization of cross attention between the encoder and decoder. The decoder can learn to attend to proper key words in the fact to predict corresponding judgement results.

Fig. 7. Point-wise mutual information (PMI) between any two categories in LJP tasks under three scenarios: (1) ground-truth data, (2) predicted results and (3) wrongly predicted results where the predicted penalty has an error of more than 1 year. Due to the large number of categories involved in our dataset, we selected top-10 law articles and top-11 (two have equal probabilities) charges with the highest co-occurrence probabilities visualization. We also divided term of penalty into 11 categories as mentioned earlier. We can see that the PMI heatmap shares the similar shape under all three scenarios. The model prediction follows the same correlation as the ground-truth data. Even when the model prediction makes significant errors, the correlation is still maintained. The strong correlation suggests the model has indeed learnt the proper dependency relationships, and we can use the dependent tasks to interpret the model predictions, e.g., the wrong prediction of the following task is because of the wrong prediction of the predecessor tasks.

4.4.5 Case Study

We provide an example predicted with our method in Figure 8. Since the defendant causes the burn of the face, neck and hands of the victim, the model makes the wrong judgment prediction which relates the case to the crime of arson. The generated court view and relevant law article content clearly explain why the model makes such a prediction. When we replace the first decoded law article with ground-truth, results of following tasks are also corrected accordingly, and the error of predicted penalty is greatly reduced from 28 months to only 2 months. The generated court view is also changed which can provide supports for the new judgment results. Similarly, the law article content also explains why the model prediction is changed from 36 months to 6 months because the penalty of arson is stronger than the penalty of intentional injury by law. The example illustrates how we can let the model “explain” itself
by decoding the whole line of tasks, which is crucial for a transparent and trustworthy LJP system.

5 RELATED WORK

Research on LJP has been studied for decades. Early researches relied primarily on hand-craft features and applied statistical [1], [2] or machine learning methods [3]–[5], [32] for it. Due to limitations of data, they focused on a tiny subset of case categories. Recently, with the accessibility of large-scale judicial datasets [21], [33], [34] and the development of deep learning [7], [7], [7], [30], [35], [36], rapid progress has been made by treating each sub-task in LJP as a text classification problem [57], [58]. Some works experimented conditioning a certain sub-task on other sub-tasks and showed improvement. For example, [7] applied reading comprehension framework to incorporate law articles for the task of predicting judgments of civil cases. [8] utilized given law articles as supplementary information to predict related charges. [9] proposed a deep gating network for charge-based prison term prediction. All these works rely on the ground-truth dependent sub-tasks, which are usually not available in real scenarios. When replacing the ground-truth labels with predicted ones, there is a significant drop of performance [12].

There have been works applying multi-task learning to share model parameters among sub-tasks [10], [11]. As all sub-tasks are classification problems, all parameters can be shared except for the task-specific prediction heads. Nevertheless, there exists a strict logical order when humans make LJP decisions, and the multi-task framework lacks the mechanism to reflect this ordered dependency. Therefore, some researchers try to establish explicit ordered dependencies among sub-tasks. For example, after encoding facts with neural network, [6] extracted top k law articles with SVM and utilized them to assist the prediction of charges. [12], [13] built dependencies among all three sub-tasks. They apply CNN for fact encoding and use LSTM to model the ordered dependencies. Each sub-task gets a task-specific representation to predict the result. In comparison, our model leverages a single unified architecture that can naturally build dependencies for arbitrary types of sub-tasks.

There have also been works focusing on court view generation [22], [39], [40], but did not leverage it to improve the LJP results. Some works utilized law article contents to assist charge prediction [41], [42]. They design complex architectures to incorporate the contents. In contrast, our framework is simple and does not affect the inference speed at all since the content is only used for training.

Leveraging a single unified text-to-text Transformer has also been applied in other NLP tasks like dialogue generation [43], [44] and question answering [45]. We adopt a similar approach in our work and further show its flexibility of enabling effective dependency learning.

6 ETHICAL STATEMENT

Since LJP is an emerging but sensitive technology, we would like to discuss ethical concerns of our work. Firstly, the corpus is created from publicly available data and the personal private information (e.g., name, plate number, etc.) has been anonymized. In addition, the proposed method aims to assist legal professionals in their research and decision-making instead of replacing them. Therefore, ethical considerations such as allowing legal rights and obligations of human beings to be decided by non-human intelligence are not breached by the system.

7 CONCLUSION

In this paper, we propose learning dependencies among sub-tasks of LJP with a unified text-to-text Transformer. It directly leverages the original legal judgment document as the prompt without handcrafting, enables full parameter sharing and supports arbitrary types and amounts of sub-tasks. We show it significantly outperforms SOTA, and the model can learn the task dependencies more effectively than previous methods. We analyze the effects of decoding order, data size and illustrate it can provide explanations to its own prediction results, even when an error is made. The proposed framework is simple and flexible. It can be easily extended to cover more legal tasks, which we leave for future work.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2016YFC0800803 ). Chuanyi Li is the corresponding author.

REFERENCES

[1] S. S. Ulmer, “Quantitative Analysis of Judicial Processes: Some Practical and Theoretical Applications,” Law and Contemporary Problems, vol. 28, no. 1, pp. 164–184, 1963, publisher: Duke University School of Law.
[2] R. Keown, “Mathematical Models for Legal Prediction,” Computer/Law journal, vol. 2, p. 829, 1980.
[3] C.-L. Liu and C.-D. Hsieh, “Exploring Phrase-Based Classification of Judicial Documents for Criminal Charges in Chinese,” in Foundations of Intelligent Systems, ser. Lecture Notes in Computer Science, F. Esposito, Z. W. Ra’s, D. Malerba, and G. Semeraro, Eds. Berlin, Heidelberg: Springer, 2006, pp. 681–690.
[4] W.-C. Lin, T.-T. Kuo, T.-J. Chang, C.-A. Yen, C.-J. Chen, and S.-D. Lin, “Exploiting Machine Learning Models for Chinese Legal Documents Labeling, Case Classification, and Sentencing Prediction,” in Proceedings of the 24th Conference on Computational Linguistics and Speech Processing (ROCLING 2012). Chung-Li, Taiwan: The Association for Computational Linguistics and Chinese Language Processing (ACL/CPLP), Sep. 2012, pp. 140–141.
[5] J. Li, G. Zhang, H. Yan, L. Yu, and T. Meng, “A Markov Logic Networks Based Method to Predict Judicial Decisions of Divorce Cases,” in 2018 IEEE International Conference on Smart Cloud (SmartCloud). Sep. 2018, pp. 129–132.
[6] B. Luo, Y. Feng, J. Xu, X. Zhang, and D. Zhao, “Learning to Predict Charges for Criminal Cases with Legal Basis,” in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2727–2736.
[7] S. Long, C. Tu, Z. Liu, and M. Sun, “Automatic Judgment Prediction via Legal Reading Comprehension,” in Chinese Computational Linguistics, ser. Lecture Notes in Computer Science, M. Sun, X. Huang, H. Ji, Z. Liu, and Y. Liu, Eds. Cham: Springer International Publishing, 2019, pp. 558–572.
[8] D. Wei and L. Lin, “An External Knowledge Enhanced Multi-label Charge Prediction Approach with Label Number Learning,” arXiv:1907.02205 [cs], Jul. 2019, arXiv: 1907.02205.
| Fact | The defendant had quarrels with the victim due to trivial matters and held grudges against him. After burning the alcohol in a tea jar, the defendant flung it at the victim, causing burns to his face, neck and back of his hands. After identification, the injury suffered by the victim was a minor injury level. After the incident, the defendant surrendered automatically. |
|---|---|
| Target | Article 234 |
| Prediction | Article 114 | × |
| Correct the article | Article 234 | Corrected |
| Charge | Crime of Intentional Injury | × |
| Penalty | 8-month imprisonment | 36-month imprisonment | Error = 28 | 6-month imprisonment | Error = 2 |
| View | The defendant intentionally burned another person with alcohol and caused a minor injury level. His behavior has constituted the crime of intentional injury. After the defendant voluntarily surrendered to the case, he truthfully confessed his criminal facts. His behavior conformed to the legal characteristics of surrendering and was given a lighter punishment in accordance with the law. | The defendant deliberately set fire, endangering public safety, and has not caused serious consequences. His behavior has constituted the crime of arson. The defendant voluntarily surrendered after committing a crime and truthfully confessed his crime, which is surrendering himself and can be punished lightly in accordance with the law. | The defendant intentionally injured the body of another person, causing minor injuries, and his behavior has constituted the crime of intentional injury. The defendant voluntarily surrendered after committing a crime and truthfully confessed his crime, which is surrendering himself and can be punished lightly in accordance with the law. |
| Content | Article 114: Whoever commits arson, breaches dikes, causes explosions, spreads pathogens of infectious diseases, poisonous or radioactive substances or other substances, or uses other dangerous means to endanger public security, but causes no serious consequences, shall be sentenced to fixed-term imprisonment of no less than three years but no more than ten years. | Article 234: Whoever intentionally injures another person is to be sentenced to not more than 3-year imprisonment. |

Fig. 8. A wrongly predicted example. When we replace the predicted law article with ground-truth, subsequent judgment results are corrected. The model-generated court view and law article content clearly explain why the model makes such a prediction.
[28] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in Proceedings of the 2016 Conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.

[29] H. Zhong, Z. Zhang, Z. Liu, and M. Sun, “Open Chinese Language Pre-trained Model Zoo,” Tech. Rep., 2019. [Online]. Available: https://github.com/thunlp/openclap

[30] J. Devlin, M.-W. Chang, K. Lee, and K. 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), 2019, pp. 4171–4186.

[31] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, “Dense passage retrieval for open-domain question answering,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6769–6781.

[32] O.-M. Süle, M. Zampieri, M. Vela, and J. van Genabith, “Predicting the Law Area and Decisions of French Supreme Court Cases,” in Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017. Varna, Bulgaria: INCOMA Ltd., Sep. 2017, pp. 716–722.

[33] I. Chalkidis, E. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos, “Large-Scale Multi-Label Text Classification on EU Legislation,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6314–6322.

[34] J. Ge, X. Shen, C. Li, and W. Hu, “Learning fine-grained fact-article correspondence in legal cases,” arXiv preprint arXiv:2104.10726, 2021.

[35] X. Shen, Y. Ouallil, C. Greenberg, M. Singh, and D. Klakow, “Estimation of gap between current language models and human performance,” Proc. Interspeech 2017, pp. 553–557, 2017.

[36] S. Qiu, B. Xu, J. Zhang, Y. Wang, X. Shen, G. De Melo, C. Long, and X. Li, “Easyaug: An automatic textual data augmentation platform for classification tasks,” in Companion Proceedings of the Web Conference 2020, 2020, pp. 249–252.

[37] X. Jiang, H. Ye, Z. Luo, W. Chao, and W. Ma, “Interpretable Rationale Augmented Charge Prediction System,” in Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations. Santa Fe, New Mexico: Association for Computational Linguistics, Aug. 2018, pp. 146–151.

[38] C. He, L. Peng, Y. Le, J. He, and X. Zhu, “SECaps: A Sequence Enhanced Capsule Model for Charge Prediction,” in Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series, ser. Lecture Notes in Computer Science, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds. Cham: Springer International Publishing, 2019, pp. 227–239.

[39] Y. Wu, K. Kuang, Y. Zhang, X. Liu, C. Sun, J. Xiao, Y. Zhuang, L. Si, and F. Wu, “De-biased court’s view generation with causality,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 763–780.

[40] Q. Li and Q. Zhang, “Court opinion generation from case fact description with legal basis,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 17, 2021, pp. 14840–14848.

[41] P. Wang, Z. Yang, S. Niu, Y. Zhang, L. Zhang, and S. Niu, “Modeling dynamic pairwise attention for crime classification over legal articles,” in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 485–494.

[42] P. Wang, P. Wang, L. Zhang, L. Shou, and W. Xu, “A recurrent attention network for judgment prediction,” in International Conference on Artificial Neural Networks. Springer, 2019, pp. 253–266.

[43] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, “A simple language model for task-oriented dialogue,” in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[44] H. Su, X. Shen, Z. Xiao, Z. Zhang, E. Chang, C. Zhang, C. Niu, and J. Zhou, “Moviechats: Chat like humans in a closed domain,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6605–6619.

[45] B. Oguz, X. Chen, V. Karpukhin, S. Peshterliev, D. Okhonko, M. Schlichtkrull, S. Gupta, Y. Mehdad, and S. Yih, “Unified open-domain question answering with structured and unstructured knowledge,” arXiv preprint arXiv:2012.14610, 2020.