Iteratively Questioning and Answering for Interpretable Legal Judgment Prediction

Haoxi Zhong,[*1] **Yuzhong Wang,**[*1] **Cunchao Tu,**[1] **Tianyang Zhang,**[2] **Zhiyuan Liu,**[†1] **Maosong Sun**[1]

[1]Department of Computer Science and Technology
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
Beijing National Research Center for Information Science and Technology, China
[2]Beijing Powerlaw Intelligent Technology Co., Ltd., China
zhonghaoxi@yeah.net, {wyzthu, tucunchao}@gmail.com, zty@powerlaw.ai, {lzy, sms}@tsinghua.edu.cn

Abstract

Legal **J**udgment **P**rediction (LJP) aims to predict judgment results according to the facts of cases. In recent years, LJP has drawn increasing attention rapidly from both academia and the legal industry, as it can provide references for legal practitioners and is expected to promote judicial justice. However, the research to date usually suffers from the lack of interpretability, which may lead to ethical issues like inconsistent judgments or gender bias. In this paper, we present QAjudge, a model based on reinforcement learning to visualize the prediction process and give interpretable judgments. QAjudge follows two essential principles in legal systems across the world: Presumption of Innocence and Elemental Trial. During inference, a Question Net will select questions from the given set and an Answer Net will answer the question according to the fact description. Finally, a Predict Net will produce judgment results based on the answers. Reward functions are designed to minimize the number of questions asked. We conduct extensive experiments on several real-world datasets. Experimental results show that QAjudge can provide interpretable judgments while maintaining comparable performance with other state-of-the-art LJP models. The codes can be found from https://github.com/thunlp/QAjudge.

Introduction

Legal **J**udgement **P**rediction (LJP) focuses on predicting judgment results (including crimes, relevant articles, and length of sentence) from the fact description of a specific case. LJP plays a crucial role in the legal system. As most people are not familiar with legal provisions, LJP can serve as legal aid by providing predicted judgment results. Besides, an LJP model with reasonable performance can provide references to professionals to save their time.

Due to its importance, many researchers pay attention to improving the performance of LJP. Early work mostly focused on extracting patterns from the fact descriptions and predict judgment results based on such patterns (Ulmer 1963; Keown 1980; Segal 1984; Lauderdale and Clark 2012; Liu, Chang, and Ho 2004; Liu and Hsieh 2006). With the recent development of deep learning, many researchers began to formalize LJP as a text classification task in **N**atural **L**anguage **P**rocessing (NLP). They employed complex deep learning models or utilized inherent properties of the LJP task (Liu, Chen, and Ho 2015; Katz, Bommarito II, and Blackman 2017; Luo et al. 2017; Ye et al. 2018; Hu et al. 2018b; Zhong et al. 2018). Existing work can already achieve a macro F1 value over 90 on charge prediction or relevant articles prediction, which is similar to that of professional judges.

Fact Description: After drinking alcohol, Bob was arrested by the police when driving a small car to the entrance of the expressway. After examination, the blood alcohol content of Bob was 173mg/100ml.	
1. Is the case related to traffic?	✓
2. Did an accident occur?	×
3. Did the party drink alcohol?	✓
Judgment Results: Reckless Driving, Article 133.	

Table 1: A judging process of a real case following the principle of Element Trial. All shown examples are translated from Chinese for illustration.

However, there are several critical shortcomings of existing methods. (1) **Noninterpretability**. Although existing LJP models has promising performance, the judging process of models is hard to interpret, which renders the judgment results unreliable. As a result, those who are unfamiliar with legal provisions cannot trust the results, while professionals will feel confused if the models give unexpected results. This makes it hard to determine whether the model has made a mistake. (2) **Ethical issues**. One of the most important issues in the judicial system is fairness, but in the real world there has always been cases unfairly judged due to discrimination based on gender or race during the trial. It is worrying that deep learning methods may learn such bias from historical judgment results and bring unfairness to the legal system (Grgic-Hlaca et al. 2018).

As a result, we focus on how a cases be interpretably

[*]Indicates equal contribution. The order is determined by dice rolling.

[†]Corresponding author.

and fairly judged and processed. There are two important principles in legal systems across the world: **Presumption of Innocence** and **Elemental Trial** (Tadros and Tierney 2004; Cohen 1982; Quintard-Morénas 2010). Presumption of Innocence means that one is considered innocent unless proven guilty. Elemental Trial is a methodology stating that judgments must be solely based on crucial elements extracted from the fact descriptions. Elemental Trail can alleviate ethical issues in judgment. As there are no gender and ethnic information in these elements, these bias will not be led into judgment. For the case shown in Figure 1, we need to decide the crimes that Bob commits. The three questions provide a step-by-step judgment process, and finally we can derive the crime should be Reckless Driving with the answers of three questions.

Therefore, one possible method for the interpretability of LJP is first predicting relevant elements from the fact description and then use the elements to predict judgment results following the principle of Elemental Trial. Professionals can utilize the detected elements for deciding judgment results, while those unfamiliar with legal provisions can also learn from these elements to understand the prediction results better. However, directly predicing all elements cannot provide the step-by-step judgment process. Humans cannot find valuable information from thousands of elements, so it is not possible to help real world legal systems.

To address these issues, we propose a new method for LJP, QAjudge. Specifically, QAjudge will first try to detect elements in the fact description by iteratively asking questions, and then use the detected elements to predict judgment results following the two principles. To avoid predicting too many elements, we adopt reinforcement learning with a carefully crafted reward function to achieve our goal of asking the minimum amount of questions to detecting crucial elements for judgment.

We have conducted experiments on three large scale datasets to verify the effectiveness and interpretability of QAjudge. Experiment results show that QAjudge can perform comparably as state-of-the-art models while providing interpretability with minimum questions asked and following the two principles. To summarize, we make several noteworthy contributions as follows:

(1) We propose an interpretable method for LJP utilizing reinforcement learning, and we are the first to explore how to integrate the real judgment process into models. QAjudge will iteratively ask questions to detect elements in the fact description, and then predict the judgment results from the elements. The whole process of QAjudge follows the actual trial process. Following the principle of Element Trail, QAjudge can alleviate ethical issues introduced by data from historical judgments intuitively.

(2) We design a new reward function for QAjudge to learn how to ask the minimum amount of questions to detect the most important elements. With only a few questions, QAjudge can achieve a promising performance on LJP.

(3) We have manually constructed the questions and elements for several crimes in Chinese legal provisions. This resource will benefit the research of LJP.

Related Work

Deep Reinforcement Learning

Deep Reinforcement learning (DRL) is first proposed by Mnih et al. (2013) to train agents to play Atari games. There are several mainstream methods in the DRL framework including Deep Q-Network (Mnih et al. 2015) and Policy Networks (Silver et al. 2016). Besides, DRL is widely used in many NLP tasks (Wu, Li, and Wang 2018; Feng et al. 2018; Li et al. 2019; 2016; Narasimhan, Kulkarni, and Barzilay 2015; He et al. 2015; Hu et al. 2018a). These works prove the rationality and effectiveness of applying DRL to NLP tasks, which support our work on LJP.

Legal Judgment Prediction

LJP is one of the most important tasks in the legal domain, and it has been studied for decades. Due to limitations of machine learning methods, early work mostly focused on analyzing cases using mathematical or statistical methods (Ulmer 1963; Nagel 1963; Segal 1984).

With the development of deep learning technology, many researchers formalize LJP as a classification task in NLP, and try to extract task specific representation to improve the performance of LJP (Liu and Hsieh 2006; Lin, Kuo, and Chang 2012; Aletras et al. 2016; Sulea et al. 2017). Besides, some researchers explore how to combine NLP technology and the prior knowledge of LJP. Luo et al. (2017) present an attention-based model to enhance LJP using contents in Legal Provision, Hu et al. (2018b) explore how to use attribute-based models to handle infrequent and confusing charges, while Zhong et al. (2018) analyze the topological dependencies between different tasks into LJP. Ye et al. (2018) attempt to generate court views according to the facts and predicted charges. However, most existing works on LJP still lack of interpretability, which prevents the application of LJP in real-world scenarios.

Methodology

In this section, we define the task of LJP, and then introduce the details of our method, QAjudge. The overall structure of QAjudge is illustrated in Figure 1.

Task Formulation

Following the setting of most existing LJP works, we will focus on LJP in civil law. LJP takes the fact description as the input. The fact description is a character sequence $\mathbf{x} = \{x_1, x_2, \cdots, x_l\}$, where l is the number of characters. The goal of LJP aims to predict the judgment results from the fact description $\mathbf{x}$. In this paper, the judgment results may be the relevant legal articles or applicable crimes. We formalize the task of LJP as a single-label classification task. More specifically, we need to predict the corresponding judgment result $y \in Y$ where Y is the label set of our task. For example, Y may contain robbery, theft, or manslaughter when the task is predicting applicable crimes.

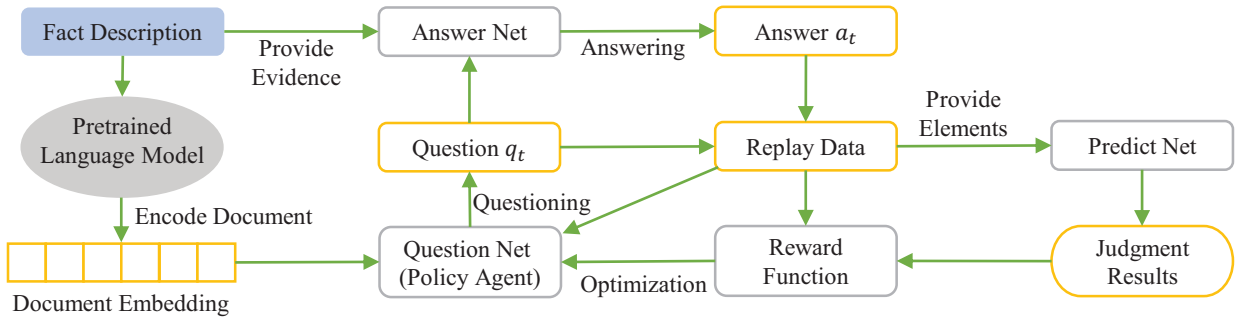

Figure 1: An overview of the framework of QAjudge.

General Framework

We will introduce the whole process of QAjudge in this part. The details of the process can be found from Figure 1. Specifically, there are several steps in QAjudge:

(1) Initialization. For initialization, QAjudge uses a pre-trained language model to encode the input fact description. Here we employ Bert (Devlin et al. 2019) as it has been fully pre-trained on large scale datasets. This gives us a document embedding $d \in \mathbb{R}^n$ where n is the dimension of the embedding. The embedding d will be fed into Question Net for selecting the most valuable question in every turn.

(2) Question Asking and Element Detection. Suppose QAjudge is restricted to ask at most K questions. At time step t, there will be $t-1$ questions which have already been asked and answered. We use (q_i, a_i) to denote the question and answer in the i-th round. Here q_i is selected from a question set Q and $a_i \in \{\text{Yes}, \text{No}\}$. This means all questions are yes-no questions as we only need to know whether the corresponding element exists. Question Net will use the document embedding d and the replay data $(q_1, a_1, q_2, a_2, \cdots, q_{t-1}, a_{t-1})$ to select a question q_t for this round. Then Answer Net will view the fact description to give out the answer a_t for question q_t.

(3) Predict Judgment Results. After asking and answering K questions, Predict Net will collect replay data $(q_1, a_1, q_2, a_2, \cdots, q_K, a_K)$ to predict the judgment results.

Question Net

Question Net F_Q in QAjudge is used for selecting question from the questions set. Suppose we have M different questions referring to M different elements in total. In round t, we define the question-state vector $s = (s_1, s_2, \cdots, s_M)$ as:

$$s_i = \begin{cases} 1 & \text{If the answer of } i\text{-th question is yes.} \\ 0 & \text{If } i\text{-th question has not been asked yet.} \\ -1 & \text{If the answer of } i\text{-th question is no.} \end{cases} \quad (1)$$

The question-state vector s records the asked questions and answers. Then the Question Net will calculate a score $\bar{\pi}_i$ of selecting question i:

$$\bar{\pi} = W_Q \cdot \begin{bmatrix} s \\ d \end{bmatrix}, \quad (2)$$

where $W_Q \in \mathbb{R}^{M \times (M+n)}$ is a transformation matrix in Question Net. To avoid asking the same question twice, we apply masked softmax to calculate the probability of selecting every question:

$$\pi_i = \frac{\exp \bar{\pi}_i}{\sum\limits_{j, s_j = 0} \exp \bar{\pi}_j} \quad (3)$$

Finally, Question Net selects $q_t = \arg\max_i(\pi_i)$ as the question for round t.

Answer Net

Answer Net F_A will answer the question q_t proposed by Question Net. Answer Net will read the fact description $\mathbf{x}$ to answer the question q_t. More specifically, there are two different forms of Answer Net:

(1) Multi-Label Classification. In this setting, we regard the Answer Net as a text classification model. The Answer Net can predict a vector $\psi \in \mathbb{R}^M$ where ψ_i denotes the probability of the answer of the i-th question being yes. When Question Net asks the i-th question, the Answer Net will answer yes if and only if $\psi_i > 0.5$.

(2) Question Answering. Another form for Answer Net is question answering. At round t, the input of Answer Net contains the fact description x and the description of question q_t. The question description is an interrogative sentence like "*Did the suspect kill someone*". Under this setting, Answer Net will read the question and answer yes or no according to the fact description.

We have conducted several experiments of these two different forms and the experimental results can be found from the Experiment Result section. We will analyze the experiment results and select one for the final experiments.

Reward Function

Following the principle of Presumption of Innocence, we should assume all elements as no if the questions have not been asked yet. Therefore, QAjudge needs to detect the elements with yes answer, as these elements can bring information for LJP. To optimize the parameters of Question Net, we need to design a reward function to achieve two goals: (1) Ask minimum questions to get valuable elements. (2)

Ask the question whose answers are more likely to be yes. Therefore, the designed reward function is:

$$r_t = res \times \left(1 - \alpha^{t-1}\right) \times \beta + pos_t \times \gamma - penalty_t \times \delta, \quad (4)$$

where $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ are four parameters of reward function. There are two parts of reward function, and the first part is $res \times \left(1 - \alpha^{t-1}\right) \times \beta$. Here res denotes whether the final prediction y is correct. If y is correct then $res = 1$ else $res = -1$. It means a positive reward will be given if the prediction is correct. In other words, the first part of reward function is an evaluation of final prediction.

The second part of reward function is $pos_t \times \gamma - penalty_t \times \delta$. If the answer a_t of round t is yes, then $pos_t = 1$, otherwise $pos_t = 0$. It means that there will be a positive reward if Question Net finds a question with answer yes. Moreover, we define $penalty_t$ as:

$$penalty_t = \begin{cases} 1 - 0.5^{unknown} & \text{If } pos_t = 0. \\ 0 & \text{Otherwise} \end{cases} \quad (5)$$

Here $unknown$ is the number of questions whose answers are yes and they have not been detected when K questions are asked. If $pos_t = 0$, the penalty will be larger with a larger value of $unknown$, while the gradient of penalty is decreasing with the increasing of $unknown$. So $penalty_t \times \delta$ is a penalty for the Question Net asking unsuitable questions. Combined with the positive reward $pos_t \times \gamma$, the second part of reward function is trying to minimize the number of questions asked by Question Net. Our experiment results show that the reward function has achieved these two goals in the section of Experiment Results.

Predict Net

Predict Net F_P play the role of giving the judgment results finally. The input of Predict Net is a vector p defined as:

$$p_i = \begin{cases} 1 & \text{If the answer of } i\text{-th question is yes.} \\ -1 & \text{Otherwise} \end{cases} \quad (6)$$

We regard unasked questions as having a negative answer, following the principle of Presumption of Innocence. The vector p has a dimension of M, with only -1 or 1 as values, so complex deep learning models should not be applied here. We conducted experiments using various machine learning models, and select the best one for final experiments.

Training

Three parts of QAjudge require training: Question Net, Answer Net, and Predict Net. For Predict Net, we will follow the training methods of specific machine learning models. For training Answer Net, we use the cross-entropy loss for optimization as it can be formalized as a classification task.

We employ Deep-Q-Learning to train Question Net. Specifically, we define the action-value function as

$$Q(s(t), d, a; W_Q) = \left(W_Q \cdot \begin{bmatrix} s(t) \\ d \end{bmatrix}\right)_a. \quad (7)$$

Algorithm 1 Training the Question Net

```
D ← {}.
while Training do
    Run QAjudge with input x while the prediction is y.
    for t ← 1 to K do
        D ← D ∪ {(s(t), q_t, r_t, s(t+1), d)}.
    end for
    Sample a mini-batch from D, and update W_Q with
    loss L(W_Q) in Eq. 8 using the mini-batch.
end while
```

Here $s(t)$ is the score vector in round t as defined in Eq. 2, and a is the index of selected question. So $Q(s(t), d, a; W_Q)$ denotes the score for the a-th question in round t. To optimize the score of selected question q_t, we use MSE as the loss function and optimize the parameters as Eq. 8.

$$\begin{aligned} target_t &= r_t + \epsilon \max_{q'} Q(s(t+1), d, q'; W_Q) \\ L(W_Q) &= \left(target_t - Q(s(t), d, q_t; W_Q)\right)^2 \end{aligned} \quad (8)$$

Here $target_t$ is the expected score for question q_t in round t. The first part of $target_t$ is the reward r_t, and the second part is an estimate of the future influence if question q_t is asked in this round. Then we can calculate the loss $L(W_Q)$ to evaluate the difference between the actual and expected scores, and use gradient descent to update the parameters of Question Net. The whole optimization step of Question Net is described in Algorithm 1.

Experiments

To evaluate the performance and effectiveness of QAjudge, we conduct a series of experiments on several large-scale LJP datasets. We will introduce the details and perform analyses on results in this section.

Dataset Construction

Following the task settings of Zhong et al. (2018), we select three different LJP datasets, namely CJO, PKU, and CAIL for experiments. CJO contains the legal documents published by the Chinese government on China Judgment Online[1], while PKU contains the legal documents collected from Peking University Law Online[2]. Moreover, CAIL is a LJP competition constructed by Xiao et al. (2018) with hundreds of participants.

As QAjudge needs to iteratively ask questions to detect essential elements, we select 20 major crimes and 19 major legal provisions from the datasets. For these crimes and legal provisions, we construct 29 major elements which are strongly relevant to the selected crimes and legal provisions. More details of our datasets can be found from Table 2.

[1] http://wenshu.court.gov.cn/
[2] http://www.pkulaw.com/

Datasets	CJO	PKU	CAIL
Documents	15,120	14,000	13,423
Max Length	3,025	2,756	1,544
Average Length	569.40	572.15	332.76

Table 2: Statistics of three datasets.

Baselines

To evaluate the performance of QAjudge, we implement two types of baselines for comparison. The first type is classical NLP models for classification, including:

DPCNN (Johnson and Zhang 2017). DPCNN is based on Convolutional Neural Networks. It uses region embedding layers and convolutional blocks, combined with shortcut connections proposed in ResNet(He et al. 2016). The structure of DPCNN makes deep models for NLP classification tasks possible, and it can effectively extract distant relationship features in the text.

GRU (Cho et al. 2014). GRU is a variation of LSTM (Hochreiter and Schmidhuber 1997). GRU adds a gating mechanism in the recurrent neural network unit which couples the input and forget gates to decrease the number of parameters. Moreover, it achieves similar performance as LSTM, even with fewer parameters.

Bert (Devlin et al. 2019). Bert is the model formed by multiple bidirectional Transformer (Vaswani et al. 2017) layers. The parameters of Bert has been fully pre-trained on large-scale text corpora. Recently, Bert has achieved state-of-the-art in many NLP tasks, including classification, reading comprehension, and question answering. We employ the Bert pre-trained on Chinese corpora for experiments.

The performance of these models serve as a benchmark of LJP as they are widely used in many NLP tasks. The second type of baselines are previous works designed specifically for the LJP task. These works are based on in-depth analysis of essential properties of LJP so that they can feed LJP features to deep learning models.

Fact-Law Attention Model (Luo et al. 2017). This model focuses on how to use the contents of legal provisions to predict crimes. This model first tries to predict relevant legal provisions and employs attention between the contents of legal provisions and fact descriptions to enhance the performance of charge prediction. As the model utilizes the ground truth legal provisions as prior knowledge, we cannot apply this model to predict applicable crimes.

Attribute-based Prediction Model (Hu et al. 2018b). This work observes the usefulness of attributes in the task of LJP, so they formalize LJP and the attribute prediction as a multi-task learning process. They train embeddings for attributes and use attention mechanism to find the relationship between fact description and attributes.

Topjudge (Zhong et al. 2018). This work mainly focuses on the relationship between subtasks in LJP. In a real judgment process, there exists a dependencies between different tasks in LJP. Therefore, this work formalizes the dependency as a directed acyclic graph to utilize the relationship between different tasks in LJP.

We apply these models in our experiments. Besides, for Predict Net, we will implement several machine learning methods to verify the effectiveness of these models.

Experimental Settings

To ensure a fair comparison between different models, we use trainable char-level embeddings for every model. We use Adam (Kingma and Ba 2015) to train all models except Bert, for which we use BertAdam (Devlin et al. 2019). The learning rate is 10^{-5} for Bert and 10^{-3} for all other models.

We randomly select 20% of the data as testing set. For every experiment, we compute Accuracy (Acc), Macro-Precision (MP), Macro-Recall (MR) and Macro-F_1 (MF). The size of the mini-batch is $4,096$, and hyper-parameters are tuned for different experiments. More details can be found from https://github.com/thunlp/QAjudge.

Experiment Results

There are three parts of our main experiments: Question Net, Answer Net, and Predict Net. We will show the experiment results of them seperately.

Datasets	CJO		PKU		CAIL	
Task	AC	RA	AC	RA	AC	RA
MLP	95.2	95.0	94.7	94.6	93.1	91.9
SVM	95.0	95.5	94.5	95.3	93.4	92.5
RF	**95.6**	**95.8**	94.6	95.1	93.8	92.8
LGB	95.4	95.6	**94.8**	**95.5**	**94.0**	**93.5**

Table 3: Experiment results (Macro-F_1) of Predict Net on three datasets. AC and RA represent two different task of predicting applicable crimes and relevant articles.

Predict Net The experimental results of Predict Net can be found in Table 3. The input of Predict Net is a discrete vector with dimension M, so deep learning method may not be suitable for Predict Net. Here we implement Multi-Layer Perceptron (MLP), Support Vector Machine (SVM) (Cortes and Vapnik 1995), Random Forest (RF) (Ho 1995) and Lightgbm (LGB) (Ke et al. 2017) for Predict Net. We select LGB as our Predict Net for the following experiments. From the experiment results, we observe that:

(1) Predict Net with machine learning can perform well on LJP tasks. Results show that detected elements are sufficient for LJP, and prove the effectiveness of Elemental Trial.

(2) One shortcoming of Elemental Trial is that it only uses answers for elements to predict, so certain information from the fact description will be unavoidably lost. Compared with results in Table 4 and Table 6, other baselines perform better than Predict Net. Moreover, the performance of Predict Net is an upper bound of QAjudge which limits the overall performance of QAjudge. One possible method for better performance is by introducing more relevant elements.

Answer Net The following experiments are on Answer Net, and the results can be found in Table 5. In the experiments, the model Bert-QA is the Question Answering mode

Datasets	CJO				PKU				CAIL			
Metrics	Acc	MP	MR	MF	Acc	MP	MR	MF	Acc	MP	MR	MF
DPCNN	95.8	96.0	95.9	95.8	95.9	96.1	95.8	95.8	95.1	95.3	95.3	95.2
GRU	95.7	95.7	95.8	95.7	96.1	96.1	**96.1**	96.0	95.4	95.4	95.7	95.5
Bert	96.9	**96.9**	96.9	**96.9**	96.3	96.1	96.0	96.0	**96.6**	96.1	96.3	96.2
Fact-Law	96.6	96.5	96.8	96.6	**98.4**	96.7	95.6	96.0	95.8	95.8	95.8	95.8
Attribute-based	**97.4**	96.5	**97.5**	**96.9**	98.1	**98.0**	95.2	95.9	96.5	**96.4**	**96.6**	**96.5**
Topjudge	96.8	96.7	96.6	96.6	97.7	97.5	95.7	**96.3**	96.4	96.2	96.4	96.3
QAjudge($K=3$)	88.2	88.8	88.2	87.8	88.1	89.0	88.1	88.0	88.7	89.2	88.9	88.5
QAjudge($K=6$)	92.9	93.3	92.9	92.9	92.7	93.2	92.7	92.7	91.7	92.4	91.9	91.8
QAjudge($K=9$)	94.1	94.4	94.1	94.1	93.3	93.5	93.3	93.3	92.3	92.6	92.5	92.3

Table 4: Experimental results of predicting applicable crimes.

Datasets	CJO		PKU		CAIL	
Metrics	Acc	MF	ACC	MF	ACC	MF
DPCNN	96.2	95.7	96.8	96.6	98.8	98.2
GRU	96.4	96.0	96.9	97.0	98.9	98.7
Bert	**96.7**	**96.5**	97.2	**97.2**	**99.1**	**98.9**
Atrribute-based	96.6	96.3	**97.3**	96.9	98.9	98.7
Topjudge	95.3	94.1	96.1	95.5	97.9	97.1
Bert-QA	96.6	96.2	97.0	96.8	98.8	98.6

Table 5: Experimental results of Answer Net.

of Bert. From the results, we can see that all existing methods can achieve promising performance in predicting elements. However, the model in question answering mode can be easily extended, as we only need to change the questions rather than re-train the model. As a result, we choose Bert-QA for the following experiments.

Question Net Finally, we perform the full experiments of predicting applicable crimes and relevant articles. From the results in Tables 4 and 6, we observe that:

(1) QAjudge can reach comparable performance as other state-of-the-art LJP models while providing an interpretable judgment process. On the task of predicting applicable crimes and relevant articles, the Macro-F_1 value of QAjudge is only about 4% lower than the state-of-art models, which proves the effectiveness of QAjudge.

(2) Compared with the upper bound given in Table 3, we can note that the performance of QAjudge is very close to the upper bound. The results show that Question Net in QAjudge can effectively detect the elements with positive answers. Besides, the results also show that the gap between QAjudge and the state-of-art models comes from the disadvantage of Elemental Trial, which is the information lost through Predict Net.

(3) Results show that QAjudge only needs to ask about 6 questions to achieve promising performance. This observation denotes that the designed reward function minimizes the number of questions asked, and QAjudge can be applied to real legal systems to help judges in their work.

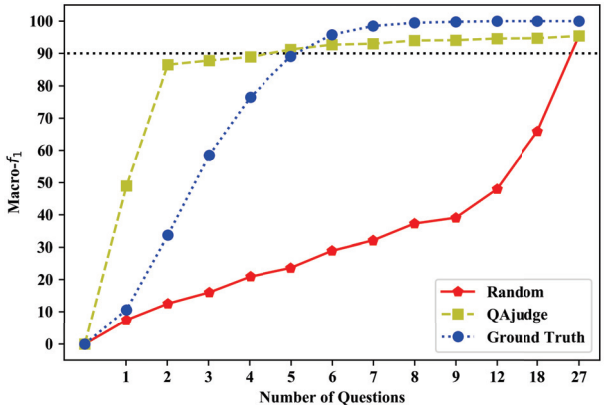

Figure 2: Experiments with different K values on the CJO.

Comparative Analysis

In this section, we analyze the relationship between the performance of QAjudge and the number of questions K. Results are summarized in Figure 2.

In Figure 2, there are three different lines: (1) **Random**. This line denotes the performance of randomly selecting K questions to ask, and then use Answer Net and Predict Net to answer. (2) **QAjudge**. The model we propose in this paper. (3) **Ground Truth**. This line denotes the percentage of data examples with at most K questions whose answers are positive. These results demonstrate the effectiveness of Question Net, as there is a huge gap between Random Ask and QAjudge. Besides, the curve of QAjudge is close to the curve of Ground Truth with increasing K, which denotes that the performance of QAjudge is approximate to the upper bound of Element Trial. To reach better performance, more relevant elements must be introduced to Predict Net.

We also experiment with different numbers of elements. In Table 7, we restrict the number of elements M to different numbers and set $K = M$ to ask all questions. We find that the more elements applied to QAjudge the better performance it can achieve. Results show that adding more elements can gain a more effective performance.

Datasets	CJO				PKU				CAIL			
Metrics	Acc	MP	MR	MF	Acc	MP	MR	MF	Acc	MP	MR	MF
DPCNN	95.7	96.3	96.3	96.2	96.0	96.6	96.3	96.3	94.5	95.7	95.2	95.2
GRU	96.0	96.4	96.6	96.4	96.0	96.4	96.3	96.3	95.6	95.7	95.7	95.7
Bert	96.8	**97.3**	**97.3**	**97.3**	**97.7**	**98.0**	**97.9**	**97.9**	96.2	**96.6**	96.6	**96.5**
Attribute-based	**96.9**	97.2	**97.3**	**97.3**	97.5	97.6	97.9	97.7	**96.5**	96.4	**96.7**	**96.5**
Topjudge	96.8	97.1	97.2	97.2	97.6	97.8	97.8	97.8	95.7	96.4	96.3	96.3
QAjudge($K = 3$)	89.1	89.2	88.8	88.6	89.7	90.3	89.4	89.5	89.4	89.4	88.3	88.2
QAjudge($K = 6$)	92.8	92.7	92.7	92.7	92.8	93.4	92.8	92.8	91.4	91.3	91.2	91.0
QAjudge($K = 9$)	93.8	93.8	93.8	93.7	93.2	93.3	93.1	93.1	93.3	92.6	93.2	92.9

Table 6: Experimental results of predicting relevant articles.

M	6	12	18	21	24	29
MF	10.5	24.0	54.6	65.8	85.7	95.4

Table 7: Experimental results of restricting the number of elements for predicting applicable crimes on CJO.

Error Type	Ratio
Incorrect Ground Truth	9%
Missing Questions	40%
Incorrect Prediction	14%
Missing Elements	37%

Table 8: Percentage of different error types.

Fact Description: One day, Bob borrowed RMB 2k from Alice using a fake reason for marriage decoration. ... After arrested, Bob has paid the money back.	
1. Whether the defendant sold something?	×
2. Whether the defendant made a fictional fact?	✓
3. Whether the defendant illegally possessed the property of others?	✓
4. Whether the defendant hurt others?	×
Judgment Results: Fraud, referring to Article 167.	

Table 9: Another example showing the step-by-step judgment process of QAjudge.

Error Analysis

Due to the limitation of Predict Net, there is still a minor gap between QAjudge and the state-of-the-art models. To analyze why QAjudge makes mistakes, we randomly select 100 questions from the CJO dataset that are incorrectly predicted by QAjudge trained with $K = 6$. Besides, we categorize the reasons for error into 4 types: (1) Incorrect Ground Truth. Some labels or the value of elements are wrong. (2) Missing Questions. Question Net fails to select critical questions, which leads to a wrong prediction. (3) Incorrect Prediction. Question Net successfully selects all valuable questions, but Predict Net predicts a wrong result. (4) Missing Elements. Some crucial elements are not included in the constructed question set. The results can be found from Table 8.

From the results, we can see that about 37% error are due to elements missing. It means that if we add these infrequent elements to QAjudge, the upper bound can be elevated and QAjudge can reach a better performance.

Conclusion

In this paper, we address the issue of lack of interpretability int existing methods of LJP, which may lead to ethical issues. We propose QAjudge, a reinforcement learning method by iteratively questioning and answering to provide interpretable results for LJP. QAjudge follows the principles of Presumption of Innocence and Elemental Trial to alleviate ethical issues. Experiment results show that QAjudge can achieve comparable performance as state-of-the-art LJP models while providing an interpretable judgment process, and the designed reward function can minimize the number of questions asked. As a result, QAjudge can be applied to the real legal system to provide a reliable reference.

In the future, we will seek to explore the following directions: (1) We will explore how to construct questions from legal provisions automatically. If we can automatically extract elements and questions from legal provisions, QAjudge can be easily applied to situations with more applicable crimes or relevant articles. (2) We will explore how to improve the performance of Predict Net. In this paper, the performance of QAjudge is limited by the performance of Predict Net, so how to overcome the disadvantages of Elemental Trail may be crucial for future work.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831900) and the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007).

References

Aletras, N.; Tsarapatsanis, D.; Preotiuc-Pietro, D.; and Lampos, V. 2016. Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science* 2.

Cho, K.; Merrienboer, B. V.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase

representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Cohen, J. A. 1982. The criminal procedure law of the people's republic of china. *The Journal of Criminal Law and Criminology*.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, 4171–4186.

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Grgic-Hlaca, N.; Redmiles, E. M.; Gummadi, K. P.; and Weller, A. 2018. Human perceptions of fairness in algorithmic decision making: A case study of criminal risk prediction. In *Proceedings of the 2018 World Wide Web Conference*, 903–912. International World Wide Web Conferences Steering Committee.

He, J.; Chen, J.; He, X.; Gao, J.; Li, L.; Deng, L.; and Ostendorf, M. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*, 770–778.

Ho, T. K. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hu, H.; Wu, X.; Luo, B.; Tao, C.; Xu, C.; Wu, W.; and Chen, Z. 2018a. Playing 20 question game with policy-based reinforcement learning. *arXiv preprint arXiv:1808.07645*.

Hu, Z.; Li, X.; Tu, C.; Liu, Z.; and Sun, M. 2018b. Few-shot charge prediction with discriminative legal attributes. In *Proceedings of COLING*, 487–498.

Johnson, R., and Zhang, T. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of ACL*, volume 1, 562–570.

Katz, D. M.; Bommarito II, M. J.; and Blackman, J. 2017. A general approach for predicting the behavior of the supreme court of the united states. *Plos one* 12(4).

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 3146–3154.

Keown, R. 1980. Mathematical models for legal prediction. *Computer/LJ* 2:829.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Lauderdale, B. E., and Clark, T. S. 2012. The supreme court's many median justices. *American Political Science Review*.

Li, J.; Monroe, W.; Ritter, A.; Jurafsky, D.; Galley, M.; and Gao, J. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1192–1202.

Li, X.; Yin, F.; Sun, Z.; Li, X.; Yuan, A.; Chai, D.; Zhou, M.; and Li, J. 2019. Entity-relation extraction as multi-turn question answering. *arXiv preprint arXiv:1905.05529*.

Lin, W.; Kuo, T. T.; and Chang, T. J. 2012. Exploiting machine learning models for chinese legal documents labeling, case classification, and sentencing prediction. In *Processdings of ROCLING*.

Liu, C., and Hsieh, C. D. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *Proceedings of ISMIS*, 681–690.

Liu, C.; Chang, C. T.; and Ho, J. H. 2004. Case instance generation and refinement for case-based criminal summary judgments in chinese. *Journal of Informationence & Engineering* 20(4):783–800.

Liu, Y.-H.; Chen, Y.-L.; and Ho, W.-L. 2015. Predicting associated statutes for legal problems. *Information Processing & Management* 51(1):194–211.

Luo, B.; Feng, Y.; Xu, J.; Zhang, X.; and Zhao, D. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of EMNLP*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Nagel, S. S. 1963. Applying correlation analysis to case prediction. *Texas Law Review* 42:1006.

Narasimhan, K.; Kulkarni, T.; and Barzilay, R. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Quintard-Morénas, F. 2010. The presumption of innocence in the french and anglo-american legal traditions. *The American Journal of Comparative Law* 58(1):107–149.

Segal, J. A. 1984. Predicting supreme court cases probabilistically: The search and seizure cases, 1962-1981. *American Political Science Review* 78(4):891–900.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484.

Sulea, O. M.; Zampieri, M.; Vela, M.; and Genabith, J. V. 2017. Exploring the use of text classi cation in the legal domain. In *Proceedings of ASAIL workshop*.

Tadros, V., and Tierney, S. 2004. The presumption of innocence and the human rights act. *The Modern Law Review* 67(3):402–434.

Ulmer, S. S. 1963. Quantitative analysis of judicial processes: Some practical and theoretical applications. *Law and Contemporary Problems* 28:164.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of NIPS*.

Wu, J.; Li, L.; and Wang, W. Y. 2018. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1252–1262.

Xiao, C.; Zhong, H.; Guo, Z.; Tu, C.; Liu, Z.; Sun, M.; Feng, Y.; Han, X.; Hu, Z.; Wang, H.; et al. 2018. Cail2018: A large-scale legal dataset for judgment prediction. *arXiv preprint arXiv:1807.02478*.

Ye, H.; Jiang, X.; Luo, Z.; and Chao, W. 2018. Interpretable charge predictions for criminal cases: Learning to generate court views from fact descriptions. In *Proceedings of NAACL*.

Zhong, H.; Zhipeng, G.; Tu, C.; Xiao, C.; Liu, Z.; and Sun, M. 2018. Legal judgment prediction via topological learning. In *Proceedings of EMNLP*, 3540–3549.