Legal Judgment Prediction via Relational Learning

Qian Dong
dongqian19@mails.ucas.ac.cn
Institute of Software, Chinese Academy of Sciences &
University of Chinese Academy of Sciences
Beijing, China

Shuzi Niu*
shuzi@iscas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

ABSTRACT

Given a legal case and all law articles, **L**egal **J**udgment **P**rediction (LJP) is to predict the case's violated articles, charges and term of penalty. Naturally, these labels are entangled among different tasks and within a task. For example, each charge is only logically or semantically related to some fixed articles. Ignoring these constraints, LJP methods would predict unreliable results. To solve this problem, we first formalize LJP as a node classification problem over a global consistency graph derived from the training set. In terms of node encoder, we utilize a masked transformer network to obtain case aware node representations consistent among tasks and discriminative within a task. In terms of node classifier, each node's label distribution is dependent on its neighbors' in this graph to achieve local consistency by relational learning. Both the node encoder and classifier are optimized by variational EM. Finally, we propose a novel measure to evaluate self-consistency of classification results. Experimental results on two benchmark datasets demonstrate that the F1 improvement of our method is about 4.8% compared with SOTA methods.

CCS CONCEPTS

• **Applied computing** → **Law**; • **Computing methodologies** → **Supervised learning by classification**; *Structured outputs*.

KEYWORDS

Statistical Relational Learning; Natural Language Processing; Legal Judgement Prediction

ACM Reference Format:
Qian Dong and Shuzi Niu. 2021. Legal Judgment Prediction via Relational Learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462931

1 INTRODUCTION

Recent artificial intelligence techniques make legal assistant systems popular, which aim to provide handy consulting services at a low cost for both professionals and non-professionals. Key to these systems, Legal Judgment Prediction (LJP) focuses on automatically deciding a legal case's judgment results based on law articles and fact descriptions. LJP is non-trivial due to complex structure of judgment results including violated articles, charges and terms. Some studies solve LJP within a multi-task learning framework by transferring representations from different tasks and introducing topological task dependencies into judgment predictor. Others focus on few-shot and confusing labels within only one task.

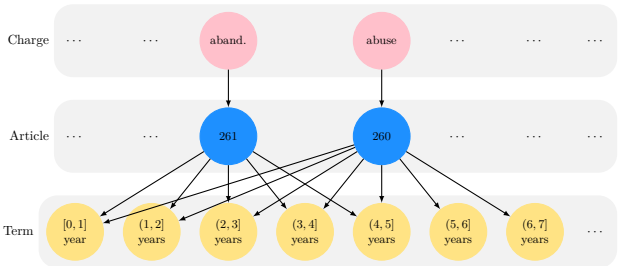


Figure 1: Illustration of Small Knowledge Graph dealing with Article 261[1] and 260[2].

Complicated judgment results intrinsically lie in perplexing class label relations. Fig. 1 only describes a subgraph of such relations. Generally there are two kinds of relations: one is within a task, the other is among different tasks. For example, *crime of abandonment* from charge prediction task and *Article 261* from article prediction task are related. Perplexing relations pose two challenges to LJP methods.

Self-consistency Challenge. Distinguished from topological task dependency, semantic or logical label relations of different judgment kinds naturally exist. For example in Fig. 1, article 261 is only related to charge "Crime of Abandonment". If a model predicts that article 261 holds, then it will be expected to output charge "Crime of Abandonment". The term mentioned in article 261 is no more than 5 years, which means that article 261 is related to first 5 term interval labels. If the predicted article label is 261, the predicted term is expected to be among these 5 labels. Existing methods have no guarantee of satisfying these constraints and logically conflicting judgment results may be obtained. This poses a self-consistency challenge to LJP approaches.

*Corresponding author

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07.
https://doi.org/10.1145/3404835.3462931

[1] Article 261: [Crime of Abandonment] Whoever refuses to fulfill his duty to support an aged person, minor, sick person or any other person who cannot live independently, if the circumstances are flagrant, shall be sentenced to fixed-term imprisonment of no more than five years, criminal detention or public surveillance.
[2] Article 260: [Crime of Abuse] Whoever maltreats a member of his family, if the circumstances are flagrant, shall be sentenced to fixed-term imprisonment of no more

Discriminative Representation Challenge. Confusing label relations within a task are common in legal judgment results, such as *Crime of Abandonment* and *Crime of Abuse* in Fig. 1, which are difficult to classify and widely studied in [7]. Confusing label relations are only a special kind of label dependency within a task. Here we focus on all possible label relations within a task to ensure they are discriminative enough in a multi-task learning framework. This poses a discriminative representation challenge to LJP approaches.

To tackle above challenges, we formalize the LJP task as a graph node classification problem. Naturally in the training set, a global consistency graph is composed of all possible class labels as graph nodes and two labels from different tasks co-occur in more than one case as an edge. Here legal judgment results include violated articles, charges and terms, and the derived graph is tripartite. Graph node classification is to find relevant nodes for each legal case in the global consistency graph. Then we introduce **R**elational learning into Trans**former** network to solve the graph node classification problem in terms of node representation learning and classification modules, referred to as **R-former**.

For the node encoder module, we design masking mechanism for transformer network to derive case aware node representation under the supervision of global consistency graph. Specifically, We use the summation of the derived case representation from vanilla transformer and each initialized node embedding as input. They are fed into a transformer with two cascaded masking matrices derived from the global consistency graph. The first matrix distillates pairwise information to learn global consistency representations among different tasks, which are further filtered by the second matrix to refine pairwise interactions to obtain discriminative representations within a task.

For the node classification module, we employ label propagation through Graph Convolution Network [10], to obtain the relevance score of each node according to its neighbors in this consistency graph for the sake of local consistency. However, exactly inferring the posterior joint relevance label distribution of all nodes is usually infeasible due to complicated relations in this tripartite graph. Therefore, we approximate it with a variational distribution by mean field method[18].

Finally, we optimize the evidence lower bound (ELBO) of the log-likelihood function of this posterior distribution within a variational EM framework in light of GMNN [19]. Experimental results [1] on two public benchmark datasets show that our proposed method R-former outperforms state-of-the-art baselines by about 4.8%. Ablation studies suggest that both node encoder for global consistency and node classifier for local consistency are essential for the performance improvement.

Our major contributions are listed as follows. (1) We define a global consistency graph derived from the training set, and formalize LJP task as a graph node classification problem. (2) Case aware node representations, which are consistent among different tasks and discriminative within a task, are learned through the masked transformer network based on this global consistency graph. (3) To achieve local consistency in the node classifier, we perform label propagation over this global consistency graph by graph neural network to estimate the posterior relevance label distribution. (4) Self-consistency measures are proposed to evaluate the consistency of classification results from different tasks.

2 RELATED WORK

Here we briefly describe existing LJP methods and other related techniques, i.e. Transformer, Graph Neural Network and statistical relational learning.

2.1 Legal Judgment Prediction Approaches

Legal judgment results are usually composed of related law articles, charges, terms and so on. The complex constituents of judgment results make Legal Judgment Prediction (LJP) task different from text classification or matching task. Some studies attempt to solve several tasks simultaneously, such as article, charge and term prediction tasks, while others focus only one of them.

Single Task Learning Method. Single judgment prediction task is often solved by text classification or matching model [31]. Early methods[13] use handcrafted features for charge prediction. Recent single task methods often attempt to solve few-shot and confusing label problems. Few-Shot [7] manually design several discriminative charge attributes as relational features for fact descriptions and charges, which provide effective information for both few-shot and confusing charges. Dynamic pairwise attention [27] utilizes pairwise attention to alleviate few-shot article problem and adopts dynamic threshold techniques to be adaptable to different article labels. An end-to-end memory network [21] is utilized to perform the charge prediction with the help of law article encoder. Sequence Enhanced Capsule model [6] utilizes a seq-caps layer and an attention residual unit for charge prediction and is optimized with focal loss to relieve few-shot charge problem.

Multi-Task Learning Framework. Multi-task learning methods focus on task dependency among charge, article and term prediction tasks. Most studies care for transferable representation learning among tasks, which is usually implemented as parameter sharing. FLA [14] solves charge prediction problem by text matching between cases and relevant articles with a supervised attention module, and optimizes charge and article prediction task jointly. LADAN [29] derives differences between confusing law articles by graph neural network to enhance the representation of fact description. Some other studies attempt to model the task dependency in the prediction stage. TopJudge [34] is an effective multi-task learning framework to model topological dependencies among subtasks with directed acyclic graphs. HMN [26] takes advantage of the hierarchical structure between charges and articles to make a correct decision. Yang et al. [30] further add backward dependencies between the prediction results of subtasks to this topological framework.

2.2 Other Related Techniques

Transformer architecture achieves great success in NLP tasks. Its self-attention mechanism expands the sequential dependency of RNN to the global dependency of each token in the input sequence on every other token in the sequence [33]. However, the

than two years, criminal detention or public surveillance.Whoever commits the crime mentioned in the preceding paragraph and causes serious injury or death to the victim shall be sentenced to fixed-term imprisonment of no less than two years but no more than seven years.

[1] The implementation has been released at https://github.com/DQ0408/R-former.

quadratic dependency on the sequence length leads to high memory requirement, which hinders these models from capturing longer context. To solve this problem, Transformer-XL [4] divides the long sequence into some overlapped subsequences and introduces relative position embedding. Sparse attention mechanisms are common methods to break down the full attention matrix [1, 33].

Graph neural network (GNN) has recently been widely studied in various areas for its ability of capturing high-order relations. It helps learn useful representations for graph node classification [3, 5, 9, 23]. The key information propagation step of GNN is to collect neighbor information to update the current node information, which encode the local graph structure and node representations. Graph Convolution Network (GCN) adopts convolution operator over the graph for information propagation, which is supposed to be effective for node classification [10]. Moreover, attention mechanism is applied to graph neural network to focus on important nodes and significant information of these nodes to improve the signal-to-noise ratio of the original data [24]. Recursive based GNN [12] attempts to use the gate mechanism like GRU [2] in the propagation step to improve the long-term propagation of information across the graph structure.

Relational Learning methods focus on modeling the joint distribution over relational data in face of the uncertainty and complex structure in real-world data [8, 11]. It does well with relational data without independent assumption. Statistical relational learning methods for relational data classification mainly model label dependency with probabilistic graphical models. Many methods adopt markov networks, such as conditional random fields [25], for relational data classification, including relational Markov networks [22] and Markov Logic networks [20]. With the development of deep learning, more and more studies focus on learning good representations for relational data. Neural Markov Logic Networks [16] and Graph Markov Neural Network [19] are two state-of-the-art methods to emphasize the representation learning. However, GMNN also concerns of the inference/prediction model with the help of label dependency. Exact inference over the posterior label distribution in probabilistic graphical models is challenging due to complex relational structure. Approximate inference methods are usually introduced, such as mean field methods [18] and belief propagation [32].

3 PROBLEM FORMALIZATION

The law article set is $\mathcal{A} = \{a_1, \ldots, a_{m_1}\}$, where m_1 is the number of law articles. The training legal case set is denoted as $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, n is the number of legal cases. $\mathbf{x}_i$ is the textual fact description and ground truth judgment results include three kinds of judgment labels, i.e. $\mathbf{y}_i = \{\mathbf{y}_i^j\}_{j=1}^{T}$. Here LJP aims at predicting each case's violated articles, charges and terms of penalty, so the number of tasks $T = 3$. Relevant articles of case c_i is $\mathbf{y}_i^1 \in \{0,1\}^{m_1}$. Charges of case c_i are represented by $\mathbf{y}_i^2 \in \{0,1\}^{m_2}$, m_2 is the number of charges, e.g."crime of intentional injury", in this law article set. All the possible terms of penalty in this law article set are divided into m_3 term intervals. Terms of case c_i is $\mathbf{y}_i^3 \in \{0,1\}^{m_3}$.

Some existing LJP approaches suppose article, charge, and term prediction tasks are independent, thus the predicted joint distribution of all tasks for case c_i is factorized as Eq.(1). $\hat{\mathbf{y}}_i$ is denoted as predicted labels of all tasks for case c_i.

$$p(\hat{\mathbf{y}}_i|\mathbf{x}_i) = \prod_{j=1}^{3}\prod_{k=1}^{m_j} P(\hat{\mathbf{y}}_i^j(k)|\mathbf{x}_i) \tag{1}$$

Several studies model the task dependency in a Directed Acyclic Graph (DAG). Thus the predicted joint distribution of all tasks for case c_i is derived as Eq.(2), where the prediction results of each task j will be conditioned on the fact representation $\mathbf{x}_i$ and outputs of all its dependent tasks D_j.

$$p(\hat{\mathbf{y}}_i|\mathbf{x}_i) = \prod_{j=1}^{3}\prod_{k=1}^{m_j} P(\hat{\mathbf{y}}_i^j(k)|\mathbf{x}_i, \{\hat{\mathbf{y}}_l\}_{l\in D_j}) \tag{2}$$

Different from task dependency in DAG, there naturally exist logic or semantic relations between different kinds of class labels. Here we model these relations derived from the training set with a tripartite graph $\mathcal{G}$ as Def. 3.1. Each class label is corresponding to a node in $\mathcal{G}$. There are three kinds of nodes $m = |\mathcal{V}| = \sum_{j=1}^{3} m_j$. All three kinds of nodes are treated equally, so we omit task subscripts. Thus for the fact description of each case $\mathbf{x}_i$ in the training set, its ground truth labels are $\mathbf{y}_i = (\mathbf{y}_i^1, \mathbf{y}_i^2, \mathbf{y}_i^3) \in \{0,1\}^m$, which means whether nodes are relevant to the case.

Definition 3.1. **Global Consistency Graph** Global Consistency Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a tripartite graph with all class labels as nodes $\mathcal{V} = \{v_k\}_{k=1}^{m}$. For each case in the training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, there is a tripartite subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$. It is derived from ground truth labels $\mathbf{y}_i$ as follows:

$$\begin{aligned}\mathcal{V}_i &= \{v_k \mid \mathbf{y}_i(k) = 1, k = 1, \ldots, m\}\\ \mathcal{E}_i &= \{(v_{k_1}, v_{k_2}) \mid v_{k_1}, v_{k_2} \in \mathcal{V}_i, k_1 \le m_1 \le k_2 \\ &\quad \text{or } m_1 \le k_1 \le m_1 + m_2 \le k_2\}\end{aligned}$$

Thus the edge set in global consistency graph $\mathcal{G}$ is computed as $\mathcal{E} = \cup_{i=1}^{n}\mathcal{E}_i$.

LJP task is reduced to a graph node classification problem over this consistency graph $\mathcal{G}$ for each case. We employ statistical relation learning to incorporate this label dependency in $\mathcal{G}$ into the classifier. Specifically the predicted joint relevance label distribution through conditional random field [25] is computed as Eq.(3), where $\phi(\cdot)$ is the potential function and $Z(\cdot)$ is the partition function.

$$p(\hat{\mathbf{y}}_i|\mathbf{x}_i, \mathcal{G}) = \frac{1}{Z(\mathbf{x}_i, \mathbf{x}_\mathcal{V})}\prod_{(v_j,v_k)\in\mathcal{E}} \phi(\hat{\mathbf{y}}_i(j), \hat{\mathbf{y}}_i(k), \mathbf{x}_i, \mathbf{x}_\mathcal{V}) \tag{3}$$

4 R-FORMER

To decrease contradictory predicted judgment results, we introduce the **R**elational learning idea into trans**former** backbone model to achieve global and local consistency, referred to as **R-former**. For the sake of global consistency, we design masking matrices based on $\mathcal{G}$, and refine node representations with masked self-attention, which are consistent among different tasks and discriminative within a task. To capture the local consistency, we predict the relevance label of one node according to its neighbors like conditional random field as p_ϕ in Eq.(3). However, exact inference of p_ϕ is intractable due to complex relations of $\mathcal{G}$. So we estimate a surrogate distribution q_θ by mean field approximation method and optimize it within a variational EM framework. Finally, we introduce self-consistency evaluation measures.

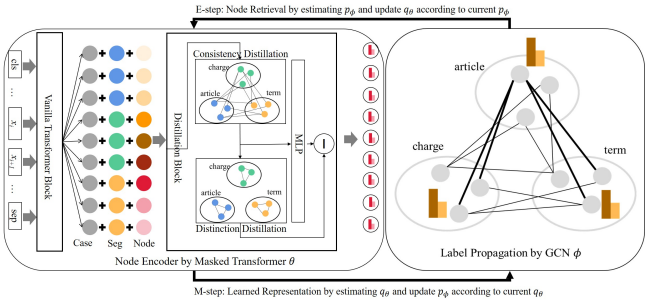


Figure 2: R-former Architecture

4.1 Architecture

As shown in Fig. 2, R-former is mainly made up of two modules: (1) Node Encoder module; (2) Node classification module. In light of GMNN [19], we alternatively optimize both modules in a variational EM framework: (1) E-step: infer relevance label distributions over $\mathcal{G}$ with p_ϕ and optimize the joint label distribution of q_θ to update θ; (2) M-step: predict relevance labels for each case with q_θ and optimize the conditional distribution p_ϕ based on predicted results to update ϕ.

Node encoder module is composed of two transformer blocks. The first is to obtain article representations from raw text. The second is used to distill consistency information among different tasks and distinction information of the same task by masking mechanisms, namely Consistency Distillation and Distinction Distillation separately in Fig. 2. All parameters in this module is denoted as θ. We obtain node representations by computing q_θ at M-step and learn this model parameter θ by maximizing the log-likelihood function of joint relevance label distribution at E-step.

The core of node classification module lies in the joint distribution $p(\hat{\mathbf{y}}_i|\mathbf{x}_i, \mathcal{G})$ in Eq.(3), which is simplified by a conditional distribution $p(\hat{\mathbf{y}}_i|\mathbf{y}_i(\mathcal{N}), \mathbf{x}_i, \mathcal{G})$ due to the independence assumption for graph $\mathcal{G}$. $\mathcal{N}$ is the corresponding neighbor set of each node in $\mathcal{G}$. Furthermore we parameterize this conditional distribution as a graph convolution neural network denoted as $p_\phi(\hat{\mathbf{y}}_i|\mathbf{y}_i(\mathcal{N}), \mathbf{x}_i, \mathcal{G})$ due to its non-linear modeling capability of label dependency. Therefore, graph convolution network is the core of node classification module in Fig. 2. We infer the relevance label distribution over $\mathcal{G}$ for each case with p_ϕ at E-step and learn ϕ by optimizing the log-likelihood of this conditional distribution at M-step.

4.2 Node Encoder Module

Taking the textual fact description $\mathbf{x}_i$ of each case c_i as input to the first vanilla transformer, we obtain the representation of the first token [cls] as the case representation denoted as $\mathbf{Q}_i \in \mathbf{R}^{1\times d}$. The node sequence $v_1, \ldots, v_k, \ldots, v_m$, $v_k \in \mathcal{V}$, is fed into the second transformer, i.e. distillation block in Fig. 2, with segment tokens representing each task, i.e. [article], [charge] and [term]. Case aware node representations $\mathbf{E}_i^0 \in \mathbf{R}^{m\times d}$ are initialized as Eq.(4), where $\mathbf{I}_k$ and $\mathbf{I}_{\text{type}(v_k)}$ are denoted as the randomly initialized embedding of v_k and its segment token type(v_k) respectively. d is the dimension of latent representations.

$$\mathbf{E}_i^0(k) = \mathbf{Q}_i + \mathbf{I}_{\text{type}(v_k)} + \mathbf{I}_k \tag{4}$$

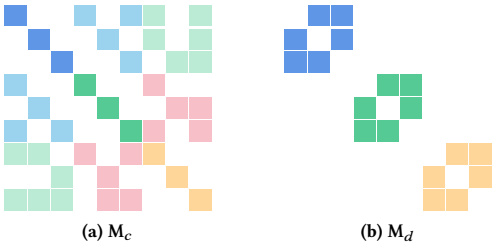
Distillation block in Fig. 2 is intrinsically L masked transformer layers. For the l-th layer, it first refine consistency information from current node representations according to consistency graph $\mathcal{G}$, referred to as consistency distillation mechanism. To achieve this goal, we design the masking matrix $\mathbf{M}_c$ according to $\mathcal{G}$ as Eq.(5) and the corresponding illustration is shown in Fig. 3(a). Through the refinement by $\mathbf{M}_c$, the masked self-attention matrix is updated as Eq.(6) to emphasize the similarity of representation between neighbor nodes in $\mathcal{G}$. ϵ is small enough.

$$\mathbf{M}_c(j,k) = \begin{cases} 1 & (v_j, v_k) \in \mathcal{E} \\ 1 & j = k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\mathbf{A}_i^l = \text{softmax}\left(\frac{(\mathbf{W}_c\mathbf{E}_i^{l-1})(\mathbf{W}_c\mathbf{E}_i^{l-1})'}{\sqrt{d}} + \epsilon(1-\mathbf{M}_c)\right) \tag{6}$$

$$\mathbf{H}_i^l(k) = \mathbf{E}_i^{l-1}(k) + \sum_j \mathbf{A}_i^l(k,j)\mathbf{E}_i^{l-1}(j) \tag{7}$$

To make node representations consistent among different tasks with $\mathcal{G}$, we update node representations $\mathbf{H}_i^l$ through masked self-attention according to Eq.(7). Each node updates its representation through its neighbor representations according to the masking matrix $\mathbf{M}_c$. For example, Article 292 is only related to the crime of assault, intentional injury, and intentional homicide. Through this tripartite masking, the model can update article node representation by its related charge node representations. Node representations without this masking mechanism would are clustered into different tasks, where node distances within a task are smaller than among different tasks. This tripartite masking mechanism attempts to zoom in related node distances among different tasks, which indirectly makes node distances within a task small enough to keep its clustered effect.



(a) $\mathbf{M}_c$ (b) $\mathbf{M}_d$

Figure 3: (a) is the tripartite masking matrix. (b) is the masking matrix for distinction distillation. The value of the white area is 0.

To learn discriminative node representations of the same task in the l-th layer, we further introduce a distinction distillation mechanism after the consistency distillation mechanism. We design a masking matrix $\mathbf{M}_d$ as Eq.(8) to depict label relations of the same task as Fig. 3(b). Pairwise similarity within each node cluster is calculated through masked self-attention mechanism as Eq.(9). Based on the pairwise similarity, similar representations are derived for nodes of the same task. Subtracting the similar representation from the learned representation $\mathbf{H}_i^l$ with consistency distillation will help

obtain a distinguishable representation as Eq.(10). This distinction distillation mechanism will pull node representations of the same task apart from each other.

$$\mathbf{M}_d(j,k) = \begin{cases} 1 & \text{type}(v_j) = \text{type}(v_k) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\hat{\mathbf{A}}_i^l = \text{softmax}(\frac{(\mathbf{W}_d\mathbf{H}_i^l)(\mathbf{W}_d\mathbf{H}_i^l)'}{\sqrt{d}} + \epsilon(1-\mathbf{M}_d)) \tag{9}$$

$$\mathbf{E}_i^l(k) = \mathbf{W}_\alpha\mathbf{H}_i^l(k) - \sum_j \hat{\mathbf{A}}_i^l(k,j)\mathbf{H}_i^l(j) \tag{10}$$

After distillation block, node encoder obtains all case aware representations $\{E_i^l\}_{l=1}^L$ over $\mathcal{G}$ given case c_i. By mean field approximation method [18], relevance labels of all nodes are predicted through a sigmoid classifier with q_θ in Eq.(11), where $\theta = \{\mathbf{W}_c, \mathbf{W}_d, \mathbf{W}_\alpha, \mathbf{W}_f\}$.

$$q_\theta(\hat{\mathbf{y}}_i|\mathbf{x}_i, \mathcal{G}) = \sigma(\mathbf{W}_f E_i^L) \tag{11}$$

Considering the uncertainty of the derivation from classification labels to relevance labels, we treat neighbors of relevant nodes in $\mathcal{G}$, whose derived labels are 0, as uncertain nodes. Thus for each case c_i, its uncertain node set is denoted as $U_i = \{v_k|\mathbf{y}_i(j) = 1, \mathbf{y}_i(k) = 0, (v_j, v_k) \in \mathcal{E}\}$. Due to these unreliable labels, we split the loss functions into two parts: $\mathcal{L}_c$ for $C_i = \mathcal{V} - U_i$, $\mathcal{L}_u$ for U_i. For nodes with precise labels, $\mathcal{L}_c$ is defined as the cross entropy between the true label distribution and the predicted distribution in Eq.(12). For nodes with uncertain labels, we approximate the true label distribution with p_ϕ and the cross entropy loss between p_ϕ and q_θ is derived as Eq.(13). To differentiate predicted results from two modules, we denote $\hat{\mathbf{y}}_i$ predicted by q_θ and $\dot{\mathbf{y}}_i$ predicted by p_ϕ.

$$\mathcal{L}_c(\mathcal{T}, \theta) = -\sum_{i=1}^{n} \sum_{v_j \in C_i} \mathbb{E}_{\mathbf{y}_i(j)} \left[\log q_\theta(\hat{\mathbf{y}}_i(j)|\mathbf{x}_i, \mathcal{G})\right] \tag{12}$$

$$\mathcal{L}_u(\mathcal{T}, \theta) = -\sum_{i=1}^{n} \sum_{v_j \in U_i} \mathbb{E}_{p_\phi(\dot{\mathbf{y}}_i(j)|\dot{\mathbf{y}}_i(\mathcal{N}_j), \mathcal{G})} \left[\log q_\theta(\hat{\mathbf{y}}_i(j)|\mathbf{x}_i, \mathcal{G})\right] \tag{13}$$

4.3 Node Classifier Module

Node classifier module is to predict the relevance label distribution over $\mathcal{G}$ given a case c_i. Due to the complicated relational structures in this consistency graph, exactly inferring the joint posterior relevance label distribution of all nodes is intractable as Eq.(3). In fact, there is no need to estimate the joint distribution, because the conditional distribution is informative enough for label dependency in $\mathcal{G}$, such as $p(\dot{\mathbf{y}}_i(j)|\mathbf{y}_i(\mathcal{N}_j), \mathbf{x}_i, \mathcal{G})$. Due to the non-linear label dependency in graph neural network, we parameterize the conditional distribution with a graph convolution network model (GCN) ϕ as $p_\phi(\dot{\mathbf{y}}_i(j)|\mathbf{y}_i(\mathcal{N}_j), \mathcal{G})$.

Specifically, we infer relevance label distributions over $\mathcal{V}$ by graph convolution network [10] with S layers. As mentioned before, there are two subsets of nodes in $\mathcal{G}$ for each case c_i: certain nodes C_i and uncertain nodes U_i. We use the ground truth label distribution per node in C_i and the predicted label distribution from q_θ per node in U_i for initialization as Eq.(14).

$$\mathbf{Y}_i^0(j) = \begin{cases} [1 - \mathbf{y}_i(j), \mathbf{y}_i(j)] & v_j \in C_i \\ q_\theta(\hat{\mathbf{y}}_i(j)|\mathbf{x}_i, \mathcal{G}) & v_j \in U_i \end{cases} \tag{14}$$

For s-th layer, graph convolution layer-wise propagation is performed by Eq.(15). In other words, we use local label dependency to estimate the current node label with its surrounding node labels at each propagation step. For better convergence, self-loop is added to $\mathcal{G}$ and its adjacency matrix $\mathbf{\Lambda}$ is symmetrically normalized based on its degree matrix $\mathbf{D}$. $\mathbf{W}_g^s$ and $\mathbf{b}_g^s$ are parameters. Through S layers, we obtain the predicted distribution as Eq.(16) for each node $v_j \in \mathcal{V}$, $\mathcal{N}_j$ is denoted as v_j's neighbor node indices.

$$\mathbf{Y}_i^s = \text{ReLU}\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{\Lambda}\mathbf{D}^{-\frac{1}{2}}\mathbf{Y}_i^{s-1}\mathbf{W}_g^s + \mathbf{b}_g^s\right) \tag{15}$$

$$p_\phi(\dot{\mathbf{y}}_i(j)|\mathbf{y}_i(\mathcal{N}_j), \mathcal{G}) = \mathbf{Y}_i^S(j) \tag{16}$$

To optimize the parameters $\phi = \{\mathbf{W}_g^s, \mathbf{b}_g^s\}_{s=1}^S$ of this classifier, we minimize the log-likelihood function of label distributions of all nodes. It is worth noting that there exist nodes whose labels are uncertain, i.e. U_i for case c_i. For those uncertain nodes $v_j \in U_i$, we use labels predicted by current q_θ as their ground truths $\mathbf{y}_i(j) = \arg\max\{q_\theta(\hat{\mathbf{y}}_i(j)|\mathbf{x}_i, \mathcal{G})\}$. Thus the optimization objective function $\mathcal{L}_a(\mathcal{T}, \phi)$ is defined as the cross entropy between ground truth and predicted label distribution in Eq.(17).

$$\mathcal{L}_a(\mathcal{T}, \phi) = -\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbb{E}_{\mathbf{y}_i(j)} \left[\log p_\phi(\dot{\mathbf{y}}_i(j)|\mathbf{y}_i(\mathcal{N}_j), \mathcal{G})\right] \tag{17}$$

Algorithm 1: Optimization Algorithm

Input: Training set $\mathcal{T}$ and Consistency graph $\mathcal{G}$
Output: model parameters θ and ϕ

% pretraining;
θ^1=arg max$_\theta$ $\mathcal{L}_c(\mathcal{T}, \theta)$ in Eq.(12);
$t = 1$;
while *convergence criteria is not achieved* **do**
  % M-step;
  for $i \in [1..n], j \in U_i$ **do**
    $\hat{\mathbf{y}}_i(j) = \arg\max_{k\in\{0,1\}}\{q_{\theta^t}(\hat{\mathbf{y}}_i(j) = k|\mathbf{x}_i, \mathcal{G})\}$ as Eq.(11);
    $\mathbf{y}_i(j) = \hat{\mathbf{y}}_i(j)$
  end
  $\phi^t = \arg\max_\phi \mathcal{L}_a(\mathcal{T}, \phi)$ as Eq.(17);
  % E-step;
  for $i \in [1..n], j \in U_i$ **do**
    compute $p_{\phi^t}(\dot{\mathbf{y}}_i(j)|\mathbf{y}_i(\mathcal{N}_j), \mathcal{G})$ as Eq.(16);
  end
  θ^{t+1}=arg max$_\theta$ $\mathcal{L}_c(\mathcal{T}, \theta) + \lambda\mathcal{L}_u(\mathcal{T}, \theta)$as Eq.(12) and (13);
  t++;
end
;

4.4 Optimization

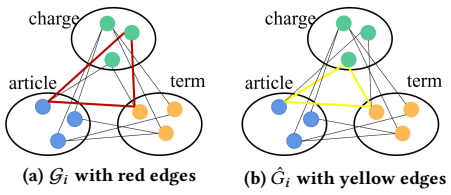
Directly optimizing the log likelihood function $\log p_\phi(\mathbf{y}_i|\mathbf{x}_i, \mathcal{G})$ in Eq.(3) is hard to obtain model parameter ϕ. So we optimize the evidence lower bound (ELBO) of $\log p_\phi(\mathbf{y}_i|\mathbf{x}_i, \mathcal{G})$ by minimizing the KL divergence between the variational distribution q_θ and target distribution p_ϕ over uncertain labels. This can be optimized by variational EM method [17].

In light of GMNN [19], the KL divergence can be optimized by alternating the following variational E-step and M-step shown in Alg. 1. At E-step, GCN with parameter ϕ is used to estimate the target label distribution p_ϕ and parameter θ from neural encoder module is learned through optimize the cross entropy between the ground truth label distribution and the predicted label distribution q_θ. Here the ground truth label distribution over uncertain data U_i is predicted with current p_ϕ.

At M-step, our proposed masked transformer blocks with parameter θ is used to infer the label distribution as q_θ. Most parameters in θ is for node representation learning, so we refer this module as node encoder. Parameter ϕ is optimized by the cross entropy between the ground truth label distribution, which is estimated with current q_θ over U_i or ground truth distribution over C_i, and predicted label distribution p_ϕ. We use transformer as backbone for node representation learning due to its superior ability in natural language understanding and long range dependency, while GMNN [19] adopts GNN as neural encoder.

4.5 Self-Consistency Evaluation

In terms of label dependency, the legal judgment results are framed as a graph. As mentioned before, the ground truth consistency graph for each case c_i is denoted as $\mathcal{G}_i$ defined in Def. 3.1, which is a subgraph of $\mathcal{G}$. Meanwhile, the predicted consistency graph for each case c_i can be derived as from predicted results $\hat{\mathbf{y}}_i$ follows: $\hat{G}_i = (\hat{V}_i, \hat{E}_i)$, $\hat{V}_i = \{v_k|\hat{\mathbf{y}}_i(k) = 1, k = 1, \ldots, m\}$, $\hat{E}_i = \{(v_j, v_k) \in \mathcal{E}|v_j, v_k \in \hat{V}_i, j \le m_1 \le k, m_1 \le j \le m_1 + m_2 \le k\}$.



(a) $\mathcal{G}_i$ with red edges (b) $\hat{G}_i$ with yellow edges

Figure 4: Comparison between ground truth consistency graph $\mathcal{G}_i$ (red edges) and predicted consistency graph $\hat{G}_i$ (yellow edges) with the background graph $\mathcal{G}$ (black edges).

Fig. 4 shows the intersection between ground truth consistency graph $\mathcal{G}_i$ and its corresponding predicted one $\hat{G}_i$ for case c_i. It is worth noting that only one relevant charge node predicted as "irrelevant" leads to two false edges. Though these two false edges are logically consistency in $\mathcal{G}$, but they are unrelated to the current case c_i. We suppose an edge to be consistent assuming two end nodes are relevant. In this sense, self consistency measure needs to be more rigorous.

Traditional classification evaluation measures, like precision, recall and F1, only focus on the node set intersection between $\hat{G}_i$ and $\mathcal{G}_i$. Based on the consistency graph Def. 3.1, node labels are used for classification accuracy performance evaluation, and edges between nodes reflect the consistency of classification results. A classifier is self-consistent if all nodes and edges between relevant nodes are classified accurately. To measure the self-consistency of a model, we estimate the Precision, Recall and Accuracy based on the edge set intersection between $\hat{G}_i$ and $\mathcal{G}_i$ for case c_i as shown in Tab. 1. F1 measure can be derived from edge precision and recall. According to measures in Tab. 1, the node precision is 2/3, but edge precision is only 1/3 in Fig. 4. We utilize these measures to evaluate self-consistency of our model on a benchmark dataset in the following section.

Table 1: Self Consistency Evaluation Measure for Classifier

	Class Acc. (Node)	Self-Consistency (edge)
Precision	$\frac{\lvert\hat{V}_i \cap \mathcal{V}_i\rvert}{\lvert V_i\rvert}$	$\frac{\lvert\hat{E}_i \cap \mathcal{E}_i\rvert}{\lvert E_i\rvert}$
Recall	$\frac{\lvert\hat{V}_i \cap \mathcal{V}_i\rvert}{\lvert\mathcal{V}_i\rvert}$	$\frac{\lvert\hat{E}_i \cap \mathcal{E}_i\rvert}{\lvert\mathcal{E}_i\rvert}$
Accuracy	$\frac{\lvert\hat{V}_i \cap \mathcal{V}_i\rvert + \lvert(\mathcal{V} - \hat{V}_i) \cap (\mathcal{V} - \mathcal{V}_i)\rvert}{\lvert\mathcal{V}\rvert}$	$\frac{\lvert\hat{E}_i \cap \mathcal{E}_i\rvert + \lvert(\mathcal{E} - \hat{E}_i) \cap (\mathcal{E} - \mathcal{E}_i)\rvert}{\lvert\mathcal{E}\rvert}$

5 EXPERIMENTS

To investigate the effectiveness and logical consistency of our proposed R-former, we conduct comprehensive experiments on two benchmark legal judgment prediction datasets.

5.1 Experimental Setting

Dataset. We use publicly available Chinese AI and Law challenge (CAIL2018) [28]: CAIL-small (the exercise stage dataset) and CAIL-big (the first stage dataset) to evaluate our method. Each sample in the data set includes a text description of the legal case and applicable laws, charges, and terms of penalty. Different data preprocessing methods are adopted in existing studies, which leads to different results. Up to the submitting deadline, LADAN is the latest LJP method as we know. To make performances from different methods comparable, we follow the data preprocessing pipeline of state-of-the-art method LADAN [29], which reproduces most state-of-the-art LJP methods with this pipeline. (1) It first filters out meaningless samples, whose legal case description has less than ten words, and complex samples with multiple relevant articles and charges. (2) It then excludes law articles and charges with less than 100 corresponding case samples. (3) It divides the terms of penalty into non-overlapping intervals. The detailed statistics of preprocessed datasets are shown in Tab. 2.

Table 2: Summary of datasets

Dataset	CAIL-small	CAIL-big
#Training Set Cases	101,619	1,587,979
#Test Set Cases	26,749	185,120
#Law Articles	103	118
#Charges	119	130
#Term of Penalty	11	11

Evaluation Metrics. Because both CAIL-small and CAIL-big are imbalanced datasets, we mainly use macro-F1 (F1) to compare with other methods. We also selected three more metrics that are widely used for multi-classification tasks, including accuracy (Acc.), macro-precision (MP), and macro-recall (MR). All these measures are evaluated at node level. To evaluate the self-consistency of LJP models, we extend these measures to the edge level as Tab. 1 and evaluate self-consistency performance on CAIL-small.

Baseline Methods Due to the complex structure of judgment results, existing state-of-the-art methods include the following three kinds: (1)Task dependency methods: **TOPJUDGE** [34] and **MPBFN-WCA** [30]. (2)Label dependency methods: **Few-Shot** [7] and **LADAN** [29]. (3)No dependency methods: **HARNN** [31] and **FLA** [14]. For the LSTM-based baseline model, we set the maximum sentence length to 100 words and the maximum document length to 15 sentences. For HARNN, FLA, Few-Shot and LADAN, we reproduce performance results with the same multi-task framework, and select the best parameter setting according to ranges from their original papers. For TOPJUDGE and MPBFN-WCA, we list their performance results reported in LADAN [29] with the same data preprocessing pipeline as our method.

Table 3: Optimizer Parameter Setting

	Pretrain θ	Opt. θ(E-step)	Opt. ϕ(M-step)
#Epochs	10	3	30
Optimizer	Adam	Adam	Adam
Learning rate	10^{-5}	10^{-6}	10^{-2}
batch size	4	4	128

Implementation Details. The distillation block of node encoder module with parameter θ is composed of $L = 3$ masked transformer layers. Graph convolution network is composed of $S = 2$ layers for node classifier module with parameter ϕ. We optimize θ for node encoder and ϕ for node classification alternatively in a variational EM framework. Both are optimized with Adam, whose parameters are set as Tab. 3. Initialized with pre-trained BERT-Base-Chinese model [2], θ is pretrained in Alg. 1. At each M-step, the optimal setting of parameter ϕ of graph convolution network for node classification is chosen as one with the smallest loss on the validation set. At E-step, the optimal θ is set to values from the epoch with the smallest loss on the validation set. The trade-off parameter λ in θ's loss function is set to 0.1. After three iterations of EM, the performance is not longer improved on the validation set in E-step. Then we end the EM loop. We show the prediction performance by q_θ on test set. All models are trained on one GTX1080Ti.

5.2 Classification Accuracy Analysis

Our proposed method R-former performs the best among all the methods in Tab. 4 on CAIL-small and CAIL-big. Compared with the best baseline LADAN, R-former's F1 improvement on CAIL-Small is 4.8%, and 5.2% and 8.7% for article, charge and term prediction task separately. Compared with LADAN, R-former's F1 improvement on CAIL-big is 6.6%, and 6.6% and 8.8% for article, charge and term prediction task separately. The superiority of LADAN to other baselines mainly lies in discriminative representation learning for confusing charges and articles, which is a kind of label dependency. This kind of label relations are solved by R-former's distinction distillation block for discriminative node representations. Besides, R-former introduces all other kinds of label dependencies into node encoder and classification modules, such as logical entailment between articles and terms.

More coarse than label dependency, task dependency is used to model the prediction order of different tasks by simulating the decision process of law professionals. This task dependency seems to have no obviously positive effect on accuracy improvement by comparing TopJUDGE and MPBFN-WCA with other methods in Tab. 4. Whether this predefined prediction order is reasonable remains a question for machine learning methods. Moreover, this prediction order easily leads to cascading errors due to unreliable result dependency on other tasks. Different from this cascaded prediction method, label propagation for optimizing R-former is mainly dependent on neighbor's ground truth label distributions.

Label dependency models, i.e. Few-shot and LADAN, for confusing labels show better performances than no dependency models, i.e. HARNN and FLA. Few-shot and LADAN's F1 performances are at least 1% higher than HARNN and FLA's for article prediction task On CAIL-small, and at least 2% higher than HARNN and FLA's for both article and charge prediction tasks on CAIL-big. In other scenarios, both kinds of methods are even. The reason lies in that confusing labels of Few-shot and LADAN are mainly from article and charge prediction tasks. Such label dependency models only obtain discriminative representations for articles and charges. In this sense, confusing label relations plays a positive role in classification accuracy performance improvement.

Compared with task dependency models mentioned above, Few-Shot and LADAN's F1 performance for article prediction is at least 1% and 3% higher on CAIL-small and CAIL-big respectively in Tab. 4. The performance improvement suggests that label dependency is directly related to classification results, though confusing label relations only directly constrain representation learning stage instead of the prediction stage. In addition to label dependency from a task, R-former also focuses on label dependency among different tasks.

The incorporation of label dependency among different tasks to representation learning stage may contribute to R-former's performance improvement compared with the best state-of-the-art baseline LADAN. Such label dependency among different tasks are also taken into account in R-former's prediction stage. So it is uncertain of which component of R-former plays a major role in the performance improvement, which will be explored in the following subsection.

5.3 Ablation Experiments

Node classifier module, consistency distillation block and distinction distillation block are three major components of R-former, denoted as p_ϕ, $\mathbf{M}_c$ and $\mathbf{M}_d$ separately. We show the performance difference between R-former without each component and R-former in Tab. 5. Three components play different roles in performance improvement in terms of different evaluation measures as shown in Tab. 5.

[2] https://github.com/google-research/bert

Table 4: Judgment prediction results on two benchmark datasets. † and ‡ means methods taking task and label dependency respectively. HARNN and FLA are pure text classification and matching models without either dependency respectively. ∗ means results reported in [29].

(a) CAIL-small

	Law Articles				Charges				Term of Penalty			
	Acc.	MP	MR	F1	Acc.	MP	MR	F1	Acc.	MP	MR	F1
HARNN	79.79	75.26	76.79	74.90	83.80	82.44	82.78	82.12	36.17	34.66	31.26	31.40
FLA	77.74	75.32	74.36	72.93	80.90	79.25	77.61	76.94	36.48	30.94	28.40	28.00
TOPJUDGE†*	79.88	79.77	73.67	73.60	82.10	83.60	78.42	79.05	36.29	34.73	32.73	29.43
MPBFN-WCA†∗	79.12	76.30	76.02	74.78	82.14	82.28	80.72	80.72	36.02	31.94	28.60	29.85
Few-Shot‡	79.30	77.80	77.59	76.09	83.65	80.84	82.01	81.55	36.52	35.07	26.88	27.14
LADAN‡∗	81.20	78.24	77.38	76.47	85.07	83.42	82.52	82.74	38.29	36.16	32.49	32.65
R-former‡	**84.48**	**82.20**	**82.67**	**81.28**	**89.13**	**88.43**	**88.00**	**87.94**	**43.77**	**42.30**	**40.94**	**41.34**

(b) CAIL-big

	Law Articles				Charges				Term of Penalty			
	Acc.	MP	MR	F1	Acc.	MP	MR	F1	Acc.	MP	MR	F1
HARNN	95.63	81.48	74.57	77.13	95.58	85.59	79.55	81.88	57.38	43.50	40.79	42.00
FLA	93.23	72.78	64.30	66.56	92.76	76.35	68.48	70.74	57.63	48.93	45.00	46.54
TOPJUDGE†∗	95.85	84.84	74.53	77.50	95.78	86.46	78.51	81.33	57.34	47.32	42.77	44.05
MPBFN-WCA†∗	96.06	85.25	74.82	78.36	95.98	89.16	79.73	83.20	58.14	45.86	39.07	41.39
Few-Shot‡	96.12	85.43	80.07	81.49	96.04	88.30	80.46	83.88	57.84	47.27	42.55	43.44
LADAN‡∗	96.57	86.22	80.78	82.36	96.45	88.51	83.73	85.35	59.66	51.78	45.34	46.93
R-former‡	**97.87**	**90.31**	**87.15**	**88.93**	**97.93**	**93.43**	**91.57**	**91.94**	**64.71**	**57.60**	**54.38**	**55.73**

Table 5: Ablation study on CAIL-small.

	Law Articles				Charges				Term of Penalty			
	Acc.	MP	MR	F1	Acc.	MP	MR	F1	Acc.	MP	MR	F1
R-former	84.48	82.20	82.67	81.28	89.13	88.43	88.00	87.94	43.77	42.30	40.94	41.34
- p_ϕ	+0.01	-0.39	-0.83	-0.69	-1.08	-1.18	-1.05	-1.3	+0.13	-1.46	-1.38	-2.57
- $\mathbf{M}_c$	-0.8	-1.38	-0.91	-1.23	-1.25	-1.37	-0.41	-1.05	0.25	-1.14	-2.42	-2.5
- $\mathbf{M}_d$	-0.45	1.07	-1.9	-0.7	-0.61	-1.03	-1.73	-1.66	-3.26	-3.01	-2.45	-4.77
- $\mathbf{M}_c$-$\mathbf{M}_d$+LADAN	-1.85	-3.67	-3.36	-3.93	-3.32	-4.89	-4.08	-4.18	-4.06	-4.47	-6.43	-6.88
LADAN	-3.28	-3.96	-4.46	-4.81	-4.06	-5.01	-5.48	-5.2	-5.48	-6.14	-8.45	-8.69
vanilla Transformer	-1.94	-0.18	-1.22	-1.47	-1.42	-0.13	-1.25	-0.94	-3.46	-2.77	-5.74	-5.98

The largest recall decrement is achieved by removing $\mathbf{M}_d$ among all three tasks in Tab. 4. In other words, the recall improvement is mainly owing to $\mathbf{M}_d$. Distinction distillation block derives similar node representations of a task through a transformer layer masked with $\mathbf{M}_d$ and subtracts such derived similar representations from original representations. This helps learn discriminative node representations within a task, which means nodes from a task are scattered more uniformly within the same area. This relaxes the decision boundary of relevant nodes, leading to more false positive nodes. Meanwhile, the number of true positive and negative nodes increases because of the accuracy improvement, which means higher recall performance.

The largest precision decrement is achieved by removing $\mathbf{M}_c$ for article and charge prediction tasks in Tab. 4. In other words, the precision improvement is mainly due to $\mathbf{M}_c$. Consistency distillation block $\mathbf{M}_c$ employs a tripartite consistency graph $\mathcal{G}$ among different tasks to ensure neighbor node representations as near as possible. The direct result of similar representations is to tighten the decision boundary of relevant nodes, which leads to less false positive nodes. Thus higher precision is achieved.

Distinction distillation block $\mathbf{M}_d$ and consistency distillation block $\mathbf{M}_c$ are complementary to the performance improvement in Tab. 5. $\mathbf{M}_d$ explores more possible relevant nodes by relaxing decision boundary within a task while $\mathbf{M}_c$ exploits label constraints among different task to remove possible irrelevant nodes by tightening the decision boundary among different tasks. This coinsides with two complementary designed masking matrices $\mathbf{M}_c$ and $\mathbf{M}_d$ in node encoder module like Fig. 3.

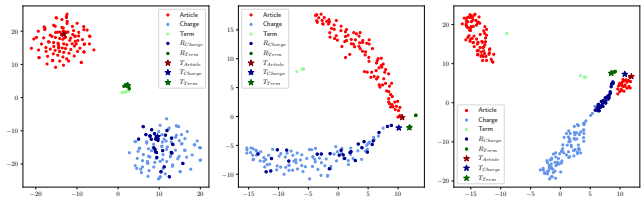
Tab. 5 suggests that p_ϕ plays an essential role in performance improvement. Compared with $\mathbf{M}_d$ for recall improvement and $\mathbf{M}_c$ for precision improvement, p_ϕ shows stable F1 improvement, which is a trade-off measure between precision and recall. p_ϕ aligns classification results among different tasks through non-linear label propagation model, and makes neighbor node classification results consistent. This adjustment of classification results is to find a trade-off boundary to satisfy as many as possible label relations.

We do ablation study by replacing q_θ with LADAN, denoted as -$\mathbf{M}_c$-$\mathbf{M}_d$+LADAN in Tab. 5. Performance differences between it and R-former in Tab. 5 suggest node encoder in R-former is more suitable for this framework than LADAN. Compared with LADAN, its performance improvement shows it is a positive example of this framework with GCN as node classifier and any model as node encoder module. Performance differences in Tab. 5 between vanilla transformer and R-former also indicate the effectiveness of R-former without vanilla transformer.

5.4 Case Study

As mentioned before, consistency distillation block makes node representations from different tasks close and consistent with $\mathcal{G}$, while distinction distillation block learns distinguishable representations for nodes within a task. To describe this insight intuitively, we choose a specific case c_i with Article 234, Charge intentional injury, Term 7 as ground truth labels, and its node representations for $\mathcal{G}$ is shown in Fig. 5 by t-sne [15] dimension reduction.

For node encoder with only $\mathbf{M}_c$, it emphasizes the similarity between neighbor nodes to keep consistent with $\mathcal{G}$, and nodes with dark colors from different tasks are closer in Fig. 5(a). Another obvious observation is that nodes are prominently clustered according to tasks. Nodes within some tasks are too close to be distinguished, such as article (red) and term prediction (green) tasks. Thus the discriminative representation challenge is left unsolved.



(a) Encoder with Only $\mathbf{M}_c$ (b) Encoder with Only $\mathbf{M}_d$ (c) Encoder with both

Figure 5: Case-aware Node Representations Learned from Node Encoder with Different Masking Matrices. Different colors means different tasks: red, green, blue are corresponding to article, term and charge prediction tasks. The ground truth labels are labeled with pentagrams. The triangle composed of three ground truth labels is denoted as $\mathcal{G}_i$. The dark green and blue colors represent nodes related to the red pentagram in global consistency graph $\mathcal{G}$.

For node encoder with only $\mathbf{M}_d$, it aims to learn discriminative representations of a task and nodes of the same color are scattered uniformly within an area in Fig. 5(b). This is in accordance with our insight. However, most dark nodes are not close enough compared with those in Fig. 5(a), which are inconsistent with $\mathcal{G}$. All dark pentagram are near to each other, which is consistent with $\mathcal{G}_i$. This local consistency is partly achieved by node classifier. In all, we need the help of $\mathbf{M}_c$ to achieve the global consistency.

For node encoder with both $\mathbf{M}_c$ and $\mathbf{M}_d$, it is to learn node representations consistent across tasks and discriminative within a task. Fig. 5(c) shows nodes with different colors are well separated and nodes of the same color are discriminative within a cluster. Such representations help achieve better classification accuracy performance. We checked the red nodes around the red pentagram, there are 15 related articles out of 26 articles for this intentional injury case. This shows that our model has the ability to inductively learn similar laws. In this sense, consistent representations are obtained with the help of both masking matrices. Moreover, we evaluate the self-consistency of a classifier with our proposed measure to verify it quantitatively in the following subsection.

5.5 Classification Consistency Analysis

Node encoder module with global consistency graph G as masking matrix is to learn consistent representations, which is qualitatively verified for a specific case in the above subsection. Node classifier module adopts graph convolution network for label propagation to make sure of local consistency of classification results, which will be quantitatively verified here. We compare R-former with LADAN and vanilla transformer as baselines methods in terms of self-consistency measures in Tab. 1 on CAIL. All the values listed in Tab. 6 are computed over all tasks.

Table 6: Self-consistency measures comparison on CAIL

	small				big			
	Edge		Node		Edge		Node	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
LADAN	48.43	65.26	67.82	76.72	72.57	84.16	83.87	85.11
BERT	50.75	67.33	70.18	81.07	74.05	85.09	86.17	87.45
R-former	54.26	70.35	72.46	82.8	75.94	86.32	86.84	88.53

Tab. 6 suggests there are fewer conflicting results among different kinds of class labels predicted by R-former. For classification accuracy performance, comparison results in Tab. 6 among different methods listed in terms of node level Accuracy and F1 are in accordance with comparison results in Tab. 4. Consistency performance comparison results among different methods agree with accuracy performance comparison results, which indicates these consistency measures are reasonable. However, consistency performances are lower than accuracy performances for each corresponding method in Tab. 6. This agrees with our definition in Tab. 1.

6 CONCLUSION

To tackle the consistency and discriminative representation challenges, we formalize the LJP task as a node classification problem over the derive global consistency graph from training data. Node representations consistent across tasks and discriminative within a task are learned through transformers masked with a tripartite graph and a task cluster graph respectively. Node classifier module predicts label distributions based on neighborhood label dependency in consistency graph through graph convolution network. Two modules are alternatively trained with a variational EM framework. Experimental results on benchmark datasets show the effectiveness and the essential role of each component. Self-consistency measure will be explored next.

ACKNOWLEDGMENTS

This research work was funded by the National Natural Science Foundation of China under Grant No.62072447.

REFERENCES

[1] Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020).

[2] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

[3] Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative Embeddings of Latent Variable Models for Structured Data. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 2702–2711.

[4] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* (2019).

[5] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., 1025–1035.

[6] Congqing He, Li Peng, Yuquan Le, Jiawei He, and Xiangyu Zhu. 2019. SECaps: A Sequence Enhanced Capsule Model for Charge Prediction. In *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis (Eds.). Springer International Publishing, Cham, 227–239.

[7] Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2018. Few-shot charge prediction with discriminative legal attributes. In *Proceedings of the 27th International Conference on Computational Linguistics*. 487–498.

[8] Hassan Khosravi and Bahareh Bina. 2010. A survey on statistical relational learning. In *Canadian conference on artificial intelligence*. Springer, 256–268.

[9] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *5th International Conference on Learning Representations* (2016).

[10] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[11] Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, Chris Meek, Jennifer Neville, et al. 2007. *Introduction to statistical relational learning*. MIT press.

[12] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.05493

[13] Chao-Lin Liu and Chwen-Dar Hsieh. 2006. Exploring Phrase-Based Classification of Judicial Documents for Criminal Charges in Chinese. In *Foundations of Intelligent Systems*, Floriana Esposito, Zbigniew W. Raś, Donato Malerba, and Giovanni Semeraro (Eds.). Springer Berlin Heidelberg, 681–690.

[14] Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to Predict Charges for Criminal Cases with Legal Basis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, Martha Palmer, Rebecca Hwa, and Sebastian Riedel (Eds.). Association for Computational Linguistics, 2727–2736. https://doi.org/10.18653/v1/d17-1289

[15] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[16] Giuseppe Marra and Ondřej Kuželka. 2019. Neural markov logic networks. *arXiv preprint arXiv:1905.13462* (2019).

[17] Radford M Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer, 355–368.

[18] Manfred Opper and David Saad. 2001. *Advanced mean field methods: Theory and practice*. MIT press.

[19] Meng Qu, Yoshua Bengio, and Jian Tang. 2019. Gmnn: Graph markov neural networks. In *International conference on machine learning*. PMLR, 5241–5250.

[20] Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning* 62, 1-2 (2006), 107–136.

[21] Yatian Shen, Jun Sun, Xiaopeng Li, Lei Zhang, Yan Li, and Xiajiong Shen. 2018. Legal Article-Aware End-To-End Memory Network for Charge Prediction *(CSAE '18)*. Association for Computing Machinery, New York, NY, USA, Article 92, 5 pages.

[22] Ben Taskar, Pieter Abbeel, Ming-Fai Wong, and Daphne Koller. 2007. Relational markov networks. *Introduction to statistical relational learning* (2007), 175–200.

[23] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[24] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[25] Hanna M Wallach. 2004. Conditional random fields: An introduction. *Technical Reports (CIS)* (2004), 22.

[26] Pengfei Wang, Yu Fan, Shuzi Niu, Ze Yang, Yongfeng Zhang, and Jiafeng Guo. 2019. Hierarchical Matching Network for Crime Classification *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 325–334.

[27] Pengfei Wang, Ze Yang, Shuzi Niu, Yongfeng Zhang, Lei Zhang, and ShaoZhang Niu. 2018. Modeling Dynamic Pairwise Attention for Crime Classification over Legal Articles *(SIGIR '18)*. Association for Computing Machinery, 485–494.

[28] Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, et al. 2018. Cail2018: A large-scale legal dataset for judgment prediction. *arXiv preprint arXiv:1807.02478* (2018).

[29] Nuo Xu, Pinghui Wang, Long Chen, Li Pan, Xiaoyan Wang, and Junzhou Zhao. 2020. Distinguish Confusing Law Articles for Legal Judgment Prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3086–3095. https://doi.org/10.18653/v1/2020.acl-main.280

[30] Wenmian Yang, Weijia Jia, Xiaojie Zhou, and Yutao Luo. 2019. Legal Judgment Prediction via Multi-Perspective Bi-Feedback Network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 4085–4091. https://doi.org/10.24963/ijcai.2019/567

[31] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. 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*. 1480–1489.

[32] Jonathan S Yedidia, William T Freeman, and Yair Weiss. 2003. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium* 8 (2003), 236–239.

[33] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html

[34] Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2018. Legal judgment prediction via topological learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3540–3549.