Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 6705405, 18 pages
https://doi.org/10.1155/2019/6705405

Research Article

RnRTD: Intelligent Approach Based on the Relationship-Driven Neural Network and Restricted Tensor Decomposition for Multiple Accusation Judgment in Legal Cases

Xiaoding Guo, Hongli Zhang, Lin Ye, and Shang Li

Harbin Institute of Technology, Harbin, China

Correspondence should be addressed to Hongli Zhang; zhanghongli@hit.edu.cn

Received 14 January 2019; Accepted 17 June 2019; Published 7 July 2019

Guest Editor: David Pokrajac

The use of intelligent judgment technology to assist in judgment is an inevitable trend in the development of judgment in contemporary social legal cases. Using big data and artificial intelligence technology to accurately determine multiple accusations involved in legal cases is an urgent problem to be solved in legal judgment. The key to solving these problems lies in two points, namely, (1) characterization of legal cases and (2) classification and prediction of legal case data. Traditional methods of entity characterization rely on feature extraction, which is often based on vocabulary and syntax information. Thus, traditional entity characterization often requires extensive energy and has poor generality, thus introducing a large amount of computation and limitation to subsequent classification algorithms. This study proposes an intelligent judgment approach called RnRTD, which is based on the relationship-driven recurrent neural network (rdRNN) and restricted tensor decomposition (RTD). We represent legal cases as tensors and propose an innovative RTD method. RTD has low dependence on vocabulary and syntax and extracts the feature structure that is most favorable for improving the accuracy of the subsequent classification algorithm. RTD maps the tensors, which represent legal cases, into a specific feature space and transforms the original tensor into a core tensor and its corresponding factor matrices. This study uses rdRNN to continuously update and optimize the constraints in RTD so that rdRNN can have the best legal case classification effect in the target feature space generated by RTD. Simultaneously, rdRNN sets up a new gate and a similar case list to represent the interaction between legal cases. In comparison with traditional feature extraction methods, our proposed RTD method is less expensive and more universal in the characterization of legal cases. Moreover, rdRNN with an RTD layer has a better effect than the recurrent neural network (RNN) only on the classification and prediction of multiple accusations in legal cases. Experiments show that compared with previous approaches, our method achieves higher accuracy in the classification and prediction of multiple accusations in legal cases, and our algorithm is more interpretable.

1. Introduction

In contemporary society, the demand for big data assistance in the judgment of legal cases, such as case intelligence research [1] and judgment [2], big data comprehensive supervision, and assistance in handling legal cases, is increasing with the development of big data and artificial intelligence technology. Researchers are committed to creating an "intelligent legal case judgment" project that combines big data and artificial intelligence. Legal case multiaccusation judgment business is an important part of the realization of such a project. Legal case multiaccusation judgment technology fully applies big data and artificial intelligence technology to service judgment making, legal case handling [3], and facilitation of the public. Big data provides judgments with recognized standards for judging legal cases and avoids the occurrence of different judgment results in similar legal cases. Artificial intelligence technology avoids the subjectivity of human beings, performs scientific and accurate analyses of cases from the perspective of cases and laws, and helps judges make objective judgment in legal cases.

The solution to using big data and artificial intelligence technology to accurately judge multiple accusations in

different legal cases involves two main points, namely, (1) construction of a comprehensive and accurate characterization method of legal cases and (2) realization of a classification and prediction algorithm for multiple accusations involved in a large number of legal cases. Figure 1 shows the process of multiaccusation classification for legal cases. Traditional methods of entity characterization are often used to model an entity by tagging it. However, these feature extraction methods are highly dependent on the vocabulary and syntax in the entity data set and require heavy manpower and material resources. The generality of the tagged model is poor. In addition, feature extraction methods based on vocabulary and syntax require strong expert knowledge as support. The resulting entity characterization considerably limits the subsequent classification algorithm, and the algorithm's accuracy becomes highly volatile.

This study proposes an intelligent legal case judgment technique called RnRTD, which is based on the relationship-driven recurrent neural network (rdRNN) and restricted tensor decomposition (RTD). Figure 2 shows the framework of our approach. We present legal case data as tensor χ and propose an RTD technique. RTD is less dependent on vocabulary and syntax than traditional feature extraction methods, and it focuses more on extracting the information of potential structures in legal case tensors. RTD maximizes the accuracy of rdRNN by combining text and structural information. RTD maps legal case tensor χ into specified feature space $\mathbb{Z}$, which decomposes the original tensor χ into core tensor $\widetilde{\chi}$ and its corresponding feature matrix set $\{C_k\}$ under the restricted condition η in RTD. The obtained core tensor $\widetilde{\chi}$ represents the tensor structure information that is most helpful in improving the accuracy of the rdRNN classification algorithm. Core tensor $\widetilde{\chi}$ can be interpreted as the most advantageous feature structure in χ for rdRNN. RTD is an important feature extraction and dimensionality reduction operation. This study uses rdRNN to update and optimize restricted condition η in RTD iteratively so that its feature space $\mathbb{Z}$ continually approaches an ideal region, thus enabling rdRNN to achieve an optimal effect in the classification of multiple accusations in legal cases.

Compared with traditional feature extraction methods, RTD obtains legal case characterization containing tensor element and structural information that is more conducive for improving the accuracy of the rdRNN classification algorithm, and it has lower dependence on vocabulary, syntax, and expert knowledge. That is, the RTD legal case characterization model has better universality and fewer requirements on the dataset format in comparison with traditional feature extraction methods. Compared with the direct use of the original legal case tensor χ as the input of the RNN classification algorithm, rdRNN with an RTD layer has a better effect on the classification of multiple accusations in legal cases. The main reason is that rdRNN constantly updates and optimizes RTD restricted condition η, thereby enabling RTD to point to feature space $\mathbb{Z}$ where rdRNN has the best effect in legal case classification.

The main contributions of this study are summarized in the following points:

(i) *This study uses a new method of characterizing legal cases.* This study expresses a legal case as a tensor and proposes an RTD method that maps the original legal case tensor into a new feature space. RTD extracts the favorable tensor structure and text information for the subsequent classification algorithm from the original legal case tensor. RTD also extracts valuable tensor features and reduces tensor dimensions. The core tensor obtained by RTD is interpreted as the most valuable tensor structure and textual feature information extracted from the original legal case tensor for the rdRNN classification algorithm.

(ii) *This study proposes rdRNN, which is a new approach for intelligent judgment of multiple accusations in legal cases.* We add a new gate and a similar case list to control the interaction between tensors of legal cases on the basis of the original neural networks. rdRNN is particularly used for the intelligent judgment of multiple accusations in legal cases. It fully considers the impact of the relationship between legal cases on the judgment results of such cases. For example, highly similar legal cases are likely to have similar judgment results and vice versa.

(iii) *This study proposes a neural network-based method for the optimization of the restricted tensor.* The restricted tensor is a bridge between the RTD algorithm and rdRNN. rdRNN controls the tensor decomposition process by optimizing the restricted tensor, which guides the core tensor along the direction that is most conducive for improving the accuracy of the classification model. We derive the partial derivative of the loss function in rdRNN for the restricted tensor and realize the optimization operation of the neural network for the restricted tensor.

Section 2 gives the recent research progress on the classification of multiple crimes in legal cases. Section 3 introduces related definitions and the concepts involved in this study. Section 4 introduces the proposed approach for the judgment of multiple accusations in legal cases. Section 5 provides the experimental results and analysis of this study, and Section 6 presents a detailed discussion of the proposed method.

2. Related Work

With the advent of the era of big data and the development of artificial intelligence technology [4], the emergence of deep neural networks provides great prospects for accurate classification and prediction [5]. Neural network-based knowledge representation and reasoning methods enable deep learning approaches to be applied to many scenarios [6]. For the legal field, the combination of artificial intelligence and law has become an inevitable trend [7]. However, current research in this area mainly focuses on

legal case modeling [8], legal case document retrieval [9], legal consultation question-and-answer systems [10], and legal case similarity reasoning work [2]. Little research has been conducted on the multiaccusation determination of cases in the legal field.

Bartolini et al. proposed a semantic annotation method for indexing and retrieving legal texts [11]. The method uses a specific segment extraction and text classification algorithm to automatically semantically mark legal documents. Aleven developed a computational model based on artificial intelligence algorithms and professional legal knowledge [2]. The model determines the correlation between cases based on the context and problem scenarios of the case. Joshi et al. proposed a text mining method for electronic evidence review of legal cases [12]. The method uses semantic topic and text classification technology to repeatedly detect the feature vocabulary in legal documents and then automatically segments and screens the documents, avoiding the manual work of legal analysts.

Sulea et al. proposed a legal case judgment system based on SVM classifier [13]. The method uses machine learning techniques to predict the legal field to which the legal case belongs and the outcome of its judgment. By accurately extracting the features of legal cases, the method can roughly predict the specific date of the case. Brninghaus and Ashley proposed a text classification method based on facts of legal cases [14]. The method uses artificial intelligence algorithms and legal background knowledge to predict the outcome of legal cases. The method extracts facts of legal cases, indexes and models them according to the features, and finally completes the classification of legal cases.

The critical part for the prediction of legal case judgments is case modeling and case classification. Traditional text modeling methods are based on feature tags, which rely heavily on the syntax and semantic information of the source data. Labeling features requires a lot of manual work and expert knowledge. Therefore, the text classification algorithm formed on this basis is not scalable, and the accuracy is highly volatile.

3. Preliminaries

This section introduces the related methods, definitions, and background knowledge involved in this study. Section 3.1 presents the basic notations and definitions. Section 3.2 provides a formal representation of the tensor decomposition problem. Section 3.3 introduces the calculation process of forward propagation in bidirectional long short-term memory (Bi-LSTM). Section 3.4 presents a formal description of the problem about intelligent legal judgment to be solved in this study.

3.1. Definitions and Notations. This section describes the relevant notations and definitions required in this work. Tensors are actually multidimensional matrices [15], which we represent in Euler script letters, such as χ and ν. We refer to the dimensions as tensor modes and to the number of a tensors modes as order. We describe the scalars in lowercase letters (such as a, b) and the vectors in boldface lowercase letters (such as c, d). We declare the matrices in capital letters, such as A and B. We use A^{T} to represent the transpose of matrix A. We express the identity matrix as I, the identity tensor as τ, and the matrix with all elements of 1 as 1. Table 1 shows all the required notations and definitions.

Definition 1 (outer product). The outer product of vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$ is denoted as $A = \mathbf{a} \circ \mathbf{b}$, where $A \in \mathbb{R}^{I \times J}$ and $A(i, j) = \mathbf{a}(i)\mathbf{b}(j)$.

Definition 2 (elementwise multiplication). The elementwise multiplication of vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^I$ is denoted as $A = \mathbf{a} * \mathbf{b}$, where $A \in \mathbb{R}^I$ and $A(i) = \mathbf{a}(i)\mathbf{b}(i)$. In another case, the elementwise multiplication of vector $\mathbf{a} \in \mathbb{R}^I$ and matrix $A \in \mathbb{R}^{I \times J}$ is denoted as $Z = \mathbf{a} * A$, where $Z \in \mathbb{R}^{I \times J}$ and $Z(i, j) = \mathbf{a}(i)A(i, j)$.

Definition 3 (Kronecker product). Given vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$, their Kronecker product is denoted as $S = \mathbf{a} \otimes \mathbf{b}$, where $S \in \mathbb{R}^{IJ}$ and $S = [\mathbf{a}(1)\mathbf{b}^{\mathrm{T}}, \mathbf{a}(2)\mathbf{b}^{\mathrm{T}}, \cdots, \mathbf{a}(I)\mathbf{b}^{\mathrm{T}}]^{\mathrm{T}}$. Given matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{P \times Q}$, their Kronecker product is denoted as $A \otimes B$.

$$A \otimes B = [A(:, 1) \otimes B(:, 1) A(:, 1) \otimes B(:, 2) \cdots A(:, J) \otimes B(:, Q-1) A(:, J) \otimes B(:, Q)]. \tag{1}$$

Definition 4 (Khatri–Rao product). Given matrices $A \in \mathbb{R}^{I \times R}$ and $B \in \mathbb{R}^{J \times R}$, their Khatri–Rao product is denoted as $A \odot B$, which is calculated by combining the Kronecker product of each corresponding column in A and B, that is

$$A \odot B = [A(:, 1) \otimes B(:, 1) \cdots A(:, r) \otimes B(:, r) \cdots A(:, R) \otimes B(:, R)]. \tag{2}$$

Definition 5 (n-mode matricization). Given an N-mode tensor χ, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. χ can be matrixed into N forms according to each mode. We denote the n-mode matricization of χ as $\chi_{(n)}$, where $\chi_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$. $\chi_{(n)}$ is obtained by keeping the nth mode unchanged while expanding and concatenating the slices of the remaining modes into a matrix.

Definition 6 (Frobenius norm of a tensor). Given an N-mode tensor χ, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the Frobenius norm of χ is denoted as

$$||\chi||_{\mathrm{F}} = \sqrt{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} \chi(i_1, i_2, \ldots, i_N)^2}. \tag{3}$$

Definition 7 (n-mode stretch). Given an N-mode tensor ν, $\nu \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, and a weight matrix W, $W \in \mathbb{R}^{J_n \times \prod_{k \neq n}^{N} J_k}$. The n-mode stretch between ν and W is expressed as $\upsilon \times^n W = \kappa$, where $\kappa \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$.

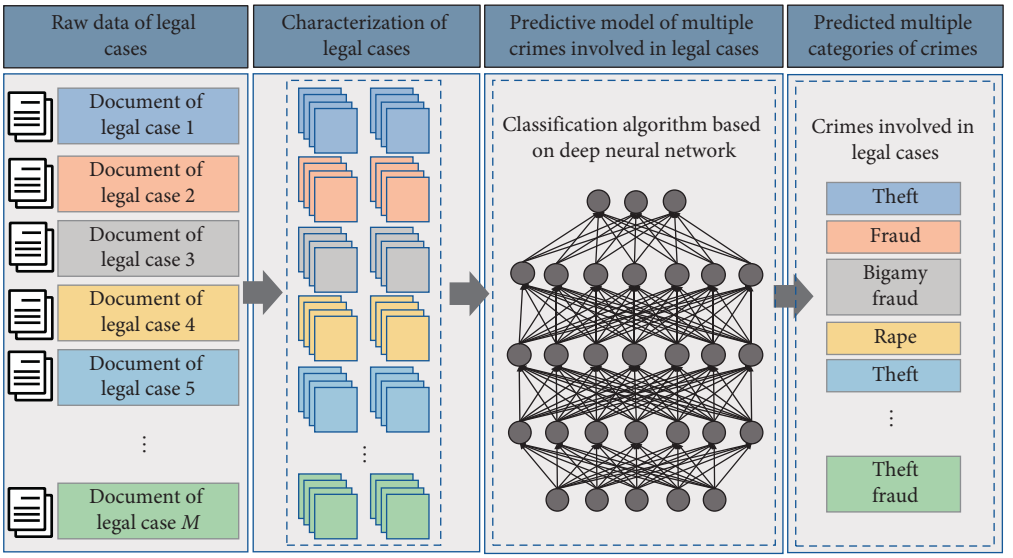

Figure 1: Multiaccusation classification for legal cases.

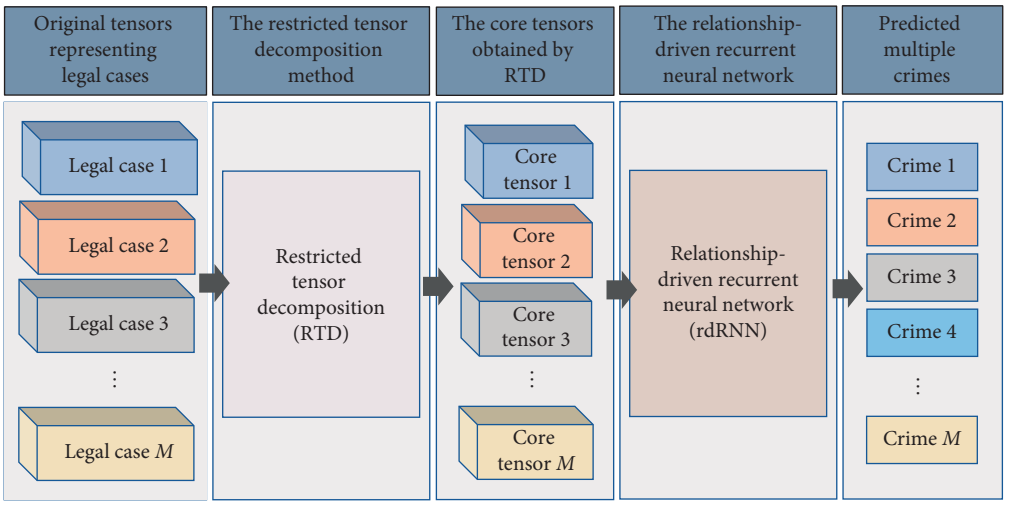

Figure 2: Framework of the proposed RnRTD.

Table 1: Symbols.

Symbol	Definition
$\chi, X, \mathbf{x}, x$	Tensor, matrix, vector, scalar
A^{T}	Transpose of matrix A
$\chi(i_1, \ldots, i_N)$	The $(i_1, \ldots, i_N)$th entry of χ, same for vectors and matrices
$A(:, i)$	The ith column of matrix A
$\circ$	Outer product
$*$	Elementwise multiplication
$\otimes$	Kronecker product
$\odot$	Khatri–Rao product
$\times^n$	The n-mode stretch
$\times_n$	The n-mode product
$\chi_{(n)}$	The n-mode matricization of tensor χ
χ_{F}	The Frobenius norm of tensor χ

$$\kappa(j_1, j_2, \ldots, j_n, \ldots, j_N) = W\left(j_1, \prod_{k \neq n}^{N} j_k\right) \upsilon(j_1, j_2, \ldots, j_n, \ldots, j_N). \tag{4}$$

Definition 8 (n-mode product). Given an N-mode tensor $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $C \in \mathbb{R}^{I_n \times J}$, their n-mode product is denoted as $\lambda = \chi \times_n C$, $\lambda \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$.

$$\lambda(i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N) = \sum_{m=1}^{I_n} \chi(i_1, \ldots, i_{n-1}, m, i_{n+1}, \ldots, i_N) C(m, j). \tag{5}$$

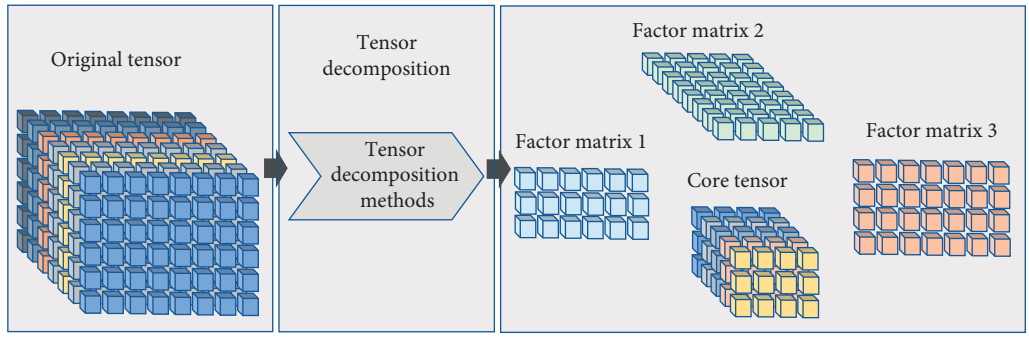

Figure 3: Tensor decomposition.

3.2. Tensor Decomposition. Many tensor decomposition methods, such as PARAFAC and Tucker decomposition, are currently available [15]. As shown in Figure 3, tensor decomposition methods decompose the original tensor into a core tensor and a series of corresponding factor matrices. The essence of tensor decomposition is to approximate the original tensor by using the product of the core tensor and the factor matrices. The mathematical description of tensor decomposition is as follows:

Given an N-mode tensor χ, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The following formula can be obtained by using the tensor decomposition method:

$$\chi \approx \tau \times_1 C_1 \times_2 C_2 \times \cdots \times_N C_N, \tag{6}$$

where τ is the core tensor, $\tau \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, and $\{C_n\}$ is the corresponding factor matrix set, $C_n \in \mathbb{R}^{J_n \times I_n}$. Each element in $\{C_n\}$ is a column orthogonal matrix. τ and $\{C_n\}$ also minimize function φ, where

$$\varphi = \left\| \chi - \tau \prod_{n}^{N} \times_n C_n \right\|_{\mathrm{F}}. \tag{7}$$

3.3. Bi-LSTM. RNNs have far-reaching implications for the study of sequence data [16]. The nodes between the hidden layers of RNN are connected [17], that is, the input of the hidden layer contains not only the output of the input layer but also the output of the hidden layer at the last moment. In theory, RNN can process sequence data of any length. However, gradient disappearance or gradient explosion often occurs when RNN deals with long-distance dependence, thereby making RNN training difficult. The hidden layers of the original RNN has only one kind of state, which is very sensitive to short-term inputs. Long short-term memory (LSTM) deals with long-distance dependence by increasing the long-term memory state in the original RNN [18].

As shown in Figure 4, we represent the input value of LSTM at time t as x_t, the output value from the previous moment $t-1$ as h_{t-1}, and the long-term unit state at time $t-1$ as c_{t-1}. We record the unit status entered at time t as $\tilde{c}_t$. The output value of LSTM at time t comprises two parts, namely, the output value of LSTM at current time h_t and the unit state of current time c_t. LSTM sets up three control gates, which are forget, input, and output, to control the long-term unit state c. The forget gate is used to determine

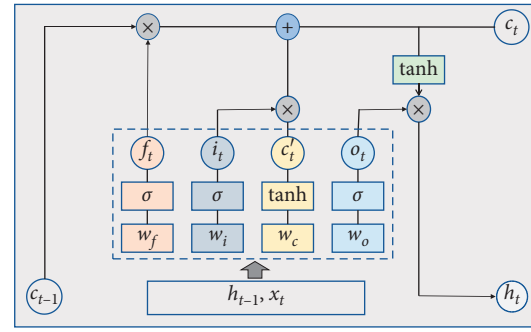

Figure 4: Framework of LSTM.

how much of the long-term unit state at the previous moment is retained at the current moment. For example, the forget gate f_t at time t determines the weight of c_{t-1} in the calculation of c_t. The input gate is used to determine how much of the input of LSTM is retained in the current long-term unit state. For example, input gate i_t determines the weight that x_t takes while calculating c_t. The output gate is used to determine how much the long-term unit state at the current moment contributes to the output of LSTM at the current time. For example, output gate o_t determines the influence of the value of c_t on h_t.

The process of forward propagation calculation in LSTM is described as follows:

$$\begin{aligned} f_t &= \sigma\left(w_f \cdot \left[h_{t-1}, x_t\right]^{\mathrm{T}} + b_f\right), \\ i_t &= \sigma\left(w_i \cdot \left[h_{t-1}, x_t\right]^{\mathrm{T}} + b_i\right), \\ \tilde{c}_t &= \tanh\left(w_c \cdot \left[h_{t-1}, x_t\right]^{\mathrm{T}} + b_c\right), \\ o_t &= \sigma\left(w_o \cdot \left[h_{t-1}, x_t\right]^{\mathrm{T}} + b_o\right). \end{aligned} \tag{8}$$

The long-term unit state c_t at current time t is calculated by f_t, i_t, c_{t-1} and $\tilde{c}_t$, and the final output of LSTM h_t is calculated by o_t and c_t. That is,

$$\begin{aligned} c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t, \\ h_t &= o_t * \tanh\left(c_t\right), \end{aligned} \tag{9}$$

where h_{t-1} is the output of LSTM at time $t-1$, x_t is the input of LSTM at time t, σ is the sigmoid function, which is our selected activation function in LSTM, $\tilde{c}_t$ is the unit state input at time t, w_f, w_i, and w_o are the weight matrices of the

forget gate f_t, the input gate i_t, and the output gate o_t, respectively, and b_f, b_i, and b_o are the bias matrices of f_t, i_t, and o_t, respectively. The activation function used in calculating $\tilde{c}_t$ is the hyperbolic tangent function, where w_c is the weight matrix and b_c is the bias term.

Bi-LSTM is a bidirectional RNN [19]. The unit state of the hidden layer in Bi-LSTM is calculated from the outputs of forward and backward LSTM. We define the output unit state of Bi-LSTM at time t as $h_{\text{Bi-LSTM}_t}$, the output unit state of forward LSTM as h_{f_t}, and the output unit state of backward LSTM as h_{b_t}. The aforementioned forward propagation formula of LSTM implies that

$$\begin{aligned} h_{f_t} &= \text{LSTM}^{\rightarrow}\left(h_{f_{t-1}}, x_t\right), \\ h_{b_t} &= \text{LSTM}^{\leftarrow}\left(h_{b_{t+1}}, x_t\right), \\ h_{\text{Bi-LSTM}_t} &= \left[h_{f_t}, h_{b_t}\right]. \end{aligned} \tag{10}$$

3.4. Problem Description

Problem 1. We express the legal case as a tensor and classify the legal case according to the judgment result. The category of each legal case is indicated by a scalar, such as r. Given a legal case dataset Ω that contains legal cases with judgment results, $\Omega = \{(\chi^{(1)}, r^{(1)}), (\chi^{(2)}, r^{(2)}), \cdots, (\chi^{(N)}, r^{(N)})\}$. $\chi^{(n)}$ represents the nth legal case in the legal case dataset Ω. $r^{(n)}$ indicates the type of legal judgment result that corresponds to the nth legal case. Our goal is to train a case classification model $\phi(\chi)$ that can classify legal cases based on their judgment results.

In this study, legal cases are represented as three-dimensional tensors. As shown in Figure 5, the first dimension represents the basic components of the case, such as the defendant's statement, the plaintiff's statement, the public prosecution, and the court's trial. On this basis, the matrix slice that contains the last two dimensions represents the matrix form of the corresponding legal case component. The matrix slice is composed of the accumulation of word vectors inside the legal case component. Generally, case components are matrixed instead of including the word vectors of all the words in the matrix. We selectively extract words that are valuable for the legal case classification. These words can be divided into two categories. The first category usually includes nouns or pronouns, such as characters, times, places, and objects; the second category usually comprises adjectives, numerals, or verbs, such as the means of committing accusations, the degree of harm, and the number of accusations.

4. Our Approach

This study proposes RnRTD for the multiaccusation determination of legal cases. Figure 6 shows the RnRTD framework. First, we extract core tensors from the original tensors using the RTD method. The core tensor approximates the restricted tensor in terms of the tensor structure and elements. Second, we use rdRNN to optimize the restricted tensor so that it guides the core tensor along the direction that is most conducive for improving the accuracy of the classification model.

4.1. RTD Method.

This study proposes a new tensor decomposition method called RTD method. The inputs of the RTD algorithm include the restricted condition tensor η and tensor χ that represents the legal case. The RTD outputs include core tensor $\tilde{\chi}$ and its corresponding factor matrix sets, namely, $\{C_k\}$ and $\{D_k\}$. RTD decomposes χ into a core tensor $\tilde{\chi}$ under the action of the restricted condition η. $\tilde{\chi}$ is approximated to η in terms of tensor structure and internal element values. RTD can be interpreted as a mapping of the original tensor χ to the core tensor $\tilde{\chi}$. In short, RTD achieves directional decomposition of tensors and extracts vital information from tensors while reducing their dimension. In this study, we define core tensor $\tilde{\chi}$ as the most favorable tensor structure and element value information for the subsequent legal case classification algorithm, namely, rdRNN. On this basis, we construct a deep neural network model for RnRTD that is dedicated to legal intelligence judgments.

RTD decomposes the original tensor under the restricted condition so that the obtained core tensor constantly approaches the restricted tensor in terms of tensor structure and element value. In Figure 7, the formal description of the problems to be solved by the RTD algorithm is shown as Problems 1 and 2.

Problem 2. Given tensor $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$, restricted tensor $\eta \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$, and its weight w_η, we derive two factor matrix sets, namely, $\{C_k\}$ and $\{D_k\}$, $C_k \in \mathbb{R}^{I_k \times H_k}$, $D_k \in \mathbb{R}^{J_k \times H_k}$, that $\{C_k\}$ and $\{D_k\}$ minimize the following function:

$$\phi = \left\| \chi \prod_{k}^{K} \times_k C_k - W_\eta \times^n \eta \prod_{k}^{K} \times_k D_k \right\|_{\text{F}}^2. \tag{11}$$

Matrix W_η is preset according to the legal case, $W_\eta \in \mathbb{R}^{H_n \times \prod_{k \neq n}^{K} H_k}$. The elements in sets $\{C_k\}$ and $\{D_k\}$ are orthogonal matrices, that is, they meet the following conditions. For any elements C_k and D_k in sets $\{C_k\}$ and $\{D_k\}$,

$$\begin{cases} C_k C_k^{\text{T}} = I, \\ C_k^{\text{T}} C_k = I, \\ D_k D_k^{\text{T}} = I, \\ D_k^{\text{T}} D_k = I. \end{cases} \tag{12}$$

In this study, we use the alternating least squares (ALS) algorithm to determine the solution of the objective function ϕ. The ALS algorithm can be divided into four steps: (1) randomly pick a variable as a parameter and randomly generate the values of other variables, (2) determine the partial derivative of the loss function ϕ in the specified parameter while fixing the values of other variables, (3) set the partial derivative of ϕ to the specified parameter as zero and calculate the value of the specified parameter, and (4) select another variable as a parameter and return to Step (2). The ALS algorithm continues to iterate Steps (2), (3),

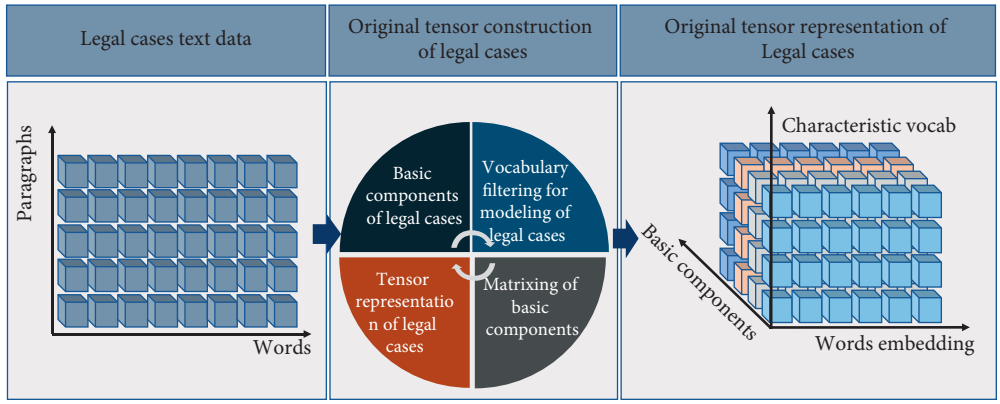

Figure 5: Original tensor that represents the legal case.

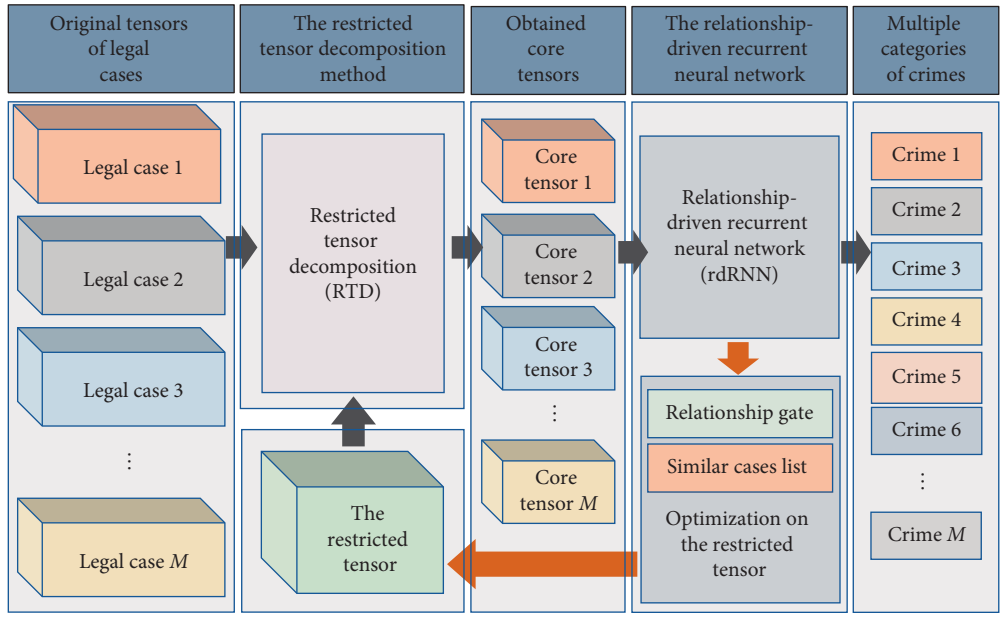

Figure 6: Framework of the RnRTD method.

and (4) until the error of the loss function ϕ reaches the tolerable upper limit.

Problem 2 needs to be solved using Lemma 1. The specific definition and proof of Lemma 1 are provided as follows.

Lemma 1. *Given function* $tr(\alpha^T\alpha) = \sum_{i_1}^{I_1}\sum_{i_2}^{I_2}\cdots\sum_{i_n}^{I_N}\alpha(i_1, i_2, \ldots, i_n)^2$, $\alpha \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, α *can be a vector, matrix, and tensor. The target function* $\varphi = tr((\chi\prod_k^K \times_k C_k)^T(\chi\prod_k^K \times_k C_k))$, *where* $\chi \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_K}$, $C_k \in \mathbb{R}^{I_k\times H_k}$, *and* C_k *satisfy equation (12). For any element* C_{k_0} *in* $\{C_k\}$, $k_0 \in [1, K]$, *the partial differential of function* φ *to* C_{k_0} *is* $\partial\varphi/\partial C_{k_0} = \varepsilon(\chi\prod_{k\neq k_0}^K \times_k C_k)^T(\chi\prod_k^K \times_k C_k)$, *where* ε *is a constant.*

The proof of Lemma 1 is shown in Proof 1.

Proof 1. We use μ to represent $\chi\prod_{k\neq k_0}^K \times_k C_k$; it can can be derived that $\varphi = \mathrm{tr}((\mu\times_{k_0} C_{k_0})^{\mathrm{T}}(\mu\times_{k_0} C_{k_0}))$. We abbreviate the target function $\varphi = \mathrm{tr}((\mu C_{k_0})^{\mathrm{T}}(\mu C_{k_0})) = \mathrm{tr}(C_{k_0}^{\mathrm{T}}\mu^{\mathrm{T}}\mu C_{k_0})$. We use ν to represent $\mu^{\mathrm{T}}\mu$, and we can get that $\varphi = \mathrm{tr}(C_{k_0}^{\mathrm{T}}\nu C_{k_0})$. According to the function derivation rule, we can obtain the following equation: $\partial\varphi/\partial C_{k_0} = \partial\mathrm{tr}(C_{k_0}^{\mathrm{T}}\nu C_{k_0})/\partial C_{k_0} = \nu C_{k_0} + \nu^{\mathrm{T}}C_{k_0}$. Since $\nu = \nu^{\mathrm{T}}$, the calculation formula for the partial derivative of the function φ to C_{k_0} is

$$\frac{\partial\varphi}{\partial C_{k_0}} = 2\nu C_{k_0} = \varepsilon\left(\chi\prod_{k\neq k_0}^{K}\times_k C_k\right)^{\mathrm{T}}\left(\chi\prod_{k}^{K}\times_k C_k\right), \tag{13}$$

where ε is a constant, $\varepsilon = 2$.

According to the iterative process of the ALS algorithm, the precondition for solving the value of $\{C_k\}$ and $\{D_k\}$ which minimize the function ϕ in equation (11) using the ALS algorithm is to calculate the value of $\partial\phi/\partial C_{k_0}$, where $k_0 \in [1, K]$. Equation (11) shows that $\partial\phi/\partial C_{k_0}$ and $\partial\phi/\partial D_{k_0}$ are solved in the same manner. Proof 2 provides mathematical proof of the calculation of $\partial\phi/\partial C_{k_0}$. □

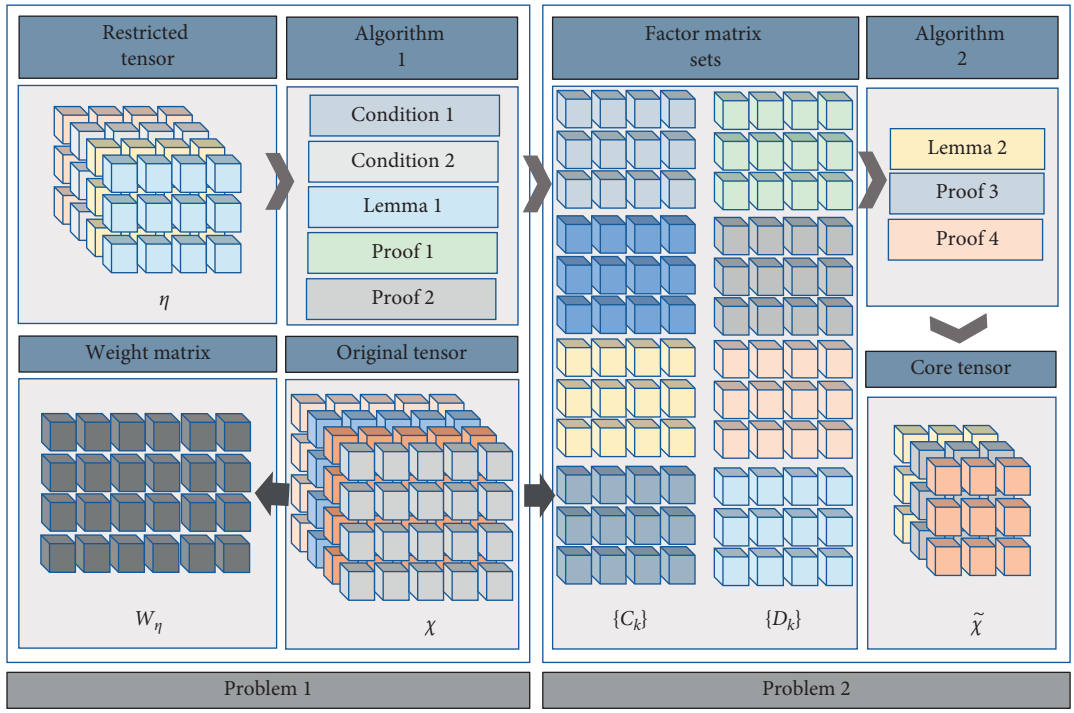

Figure 7: Framework of the RTD method.

Proof 2. We use λ and ϖ to represent $\chi\prod_{k\neq k_0}^{K}\times_k C_k$ and $w_\eta\eta\prod_k^K\times_k D_k$, respectively. According to formula (11), we can obtain that $\phi = \mathrm{tr}((\lambda\times_{k_0}C_{k_0}-\varpi)^{\mathrm{T}}(\lambda\times_{k_0}C_{k_0}-\varpi))$. We abbreviate the aforementioned formula as $\phi = \mathrm{tr}((\lambda C_{k_0}-\varpi)^{\mathrm{T}}(\lambda C_{k_0}-\varpi))$. Then, we can determine the following formula: $\partial\phi/\partial C_{k_0} = (\partial\mathrm{tr}(\varpi^{\mathrm{T}}\varpi) - 2\partial\mathrm{tr}(C_{k_0}^{\mathrm{T}}\lambda^{\mathrm{T}}\varpi) + \partial\mathrm{tr}\ (C_{k_0}^{\mathrm{T}}\lambda^{\mathrm{T}}\lambda C_{k_0}))/\partial C_{k_0}$. According to the function derivation rule, we derive that $\partial\mathrm{tr}(\varpi^{\mathrm{T}}\varpi)/\partial C_{k_0} = 0$, $(\partial\mathrm{tr}(C_{k_0}^{\mathrm{T}}\lambda^{\mathrm{T}}\varpi)/\ \partial C_{k_0}) = \lambda^{\mathrm{T}}\varpi$. In combination with Lemma 1, we can obtain that $(\partial\mathrm{tr}(C_{k_0}^{\mathrm{T}}\lambda^{\mathrm{T}}\lambda C_{k_0})/\partial C_{k_0}) = 2\lambda^{\mathrm{T}}\lambda C_{k_0}$. Finally, the following formula is determined:

$$\frac{\partial\phi}{\partial C_{k_0}} = -2\lambda^{\mathrm{T}}\varpi + 2\lambda^{\mathrm{T}}\lambda C_{k_0}. \tag{14}$$

We set the value of equation (14) to 0 and obtain that

$$\lambda C_{k_0} = \varpi. \tag{15}$$

Let $Z = \lambda_{(k_0)}^{\mathrm{T}}\varpi_{(k_0)}$, then $C_{k_0}^{\mathrm{T}}Z = (\lambda C_{k_0})_{(k_0)}^{\mathrm{T}}\varpi_{(k_0)} = \varpi_{(k_0)}^{\mathrm{T}}\varpi_{(k_0)}$. By combining Equation (12), we derive that

$$\begin{cases} C_{k_0}^{\mathrm{T}}Z = Z^{\mathrm{T}}C_{k_0}, \\ Z = C_{k_0}Z^{\mathrm{T}}C_{k_0}. \end{cases} \tag{16}$$

We use the SVD matrix decomposition method to decompose Z, and find that $Z = PSQ^{\mathrm{T}}$. P and Q are orthogonal matrices, P is a left singular matrix, Q is a right singular matrix, and S is a diagonal matrix. After this analysis, the following solution can be obtained:

$$C_{k_0} = PQ^{\mathrm{T}}. \tag{17}$$

In summary, according to equations (11)–(17), we can derive a solution of $\partial\phi/\partial C_{k_0}$ and C_{k_0} which are described in equations (14) and (17), respectively. $\partial\phi/\partial D_k$ is calculated in the same manner as $\partial\phi/\partial C_k$. On this basis, we calculate the value of $\{C_k\}$ and $\{D_k\}$, which minimize the objective function ϕ in formula (11), by using the ALS algorithm.

Algorithm 1 shows the solution of Problem 2 by using ALS algorithm. The inputs of Algorithm 1 are χ which represents one legal case, the restricted tensor η, and its weight w_η. In line 2, we randomly initialize the values of $\{C_k\}, \{U_k\}$. max_iterations in line 2 represents the maximum number of iterations of ALS algorithm. The function calcu_Z in line 4 corresponds to equation (16). Line 5 and 6 show the calculation process of equation (17).

Another problem to be solved by the RTD algorithm is Problem 3, which is the formal description of the process of tensor decomposition on the original tensor under the action of the restricted tensor and its weight. On the basis of Problem 2, we can obtain factor matrix sets $\{C_k\}$ and $\{D_k\}$, which minimize the value of function ϕ in formula (11) while satisfying formula (12). □

Problem 3. Given a tensor $\chi\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_K}$ and factor matrix sets $\{C_k\}$ and $\{D_k\}$, $C_k\in\mathbb{R}^{I_k\times H_k}$, $D_k\in\mathbb{R}^{J_k\times H_k}$, $\{C_k\}$ and $\{D_k\}$ are derived from Problem 2. A core tensor $\tilde{\chi}$ is determined, where $\tilde{\chi}\in\mathbb{R}^{J_1\times J_2\times\cdots\times J_K}$ and $\tilde{\chi}$ minimize the following target function:

$$F_{\mathrm{RTD}} = \left\|\chi\prod_k^K\times_k C_k - \tilde{\chi}\prod_k^K\times_k D_k\right\|_{\mathrm{F}}^2. \tag{18}$$

Input: Tensor χ which represents the original legal case data, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$, the restricted tensor η, $\eta \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$, and its weight w_η.
Output: The factor matrix sets $\{C_k\}$ and $\{D_k\}$, $C_k \in \mathbb{R}^{I_k \times H_k}$, $D_k \in \mathbb{R}^{J_k \times H_k}$.
Initialize the factor matrix sets $\{C_k\}$ and $\{D_k\}$;
for $i = 1$ to max_iterations **do**
  // First, pick the elements in the factor set $\{C_k\}$ as variables
  for $k_0 = 1$ to K **do**
    $Z = \text{calcu\_}Z(\chi, \eta, \{C_k\}, \{D_k\}, w_\eta, k_0)$;
    $P, S, Q^{\mathrm{T}} = \text{SVD}(Z)$;
    $C_{k_0} = PQ^{\mathrm{T}}$;
  end
  // Then, pick the elements in the factor set $\{D_k\}$ as variables
  for $k_0 = 1$ to K **do**
    $Z = \text{calcu\_}Z(\chi, \eta, \{D_k\}, \{C_k\}, w_\eta, k_0)$;
    $P, S, Q^{\mathrm{T}} = \text{SVD}(Z)$;
    $D_{k_0} = PQ^{\mathrm{T}}$;
  end
end
return $\{C_k\}$, $\{D_k\}$;

Algorithm 1: The solution of Problem 2 by using ALS Algorithm.

Problem 3 needs to be solved using Lemma 2. The specific definition and proof process of Lemma 2 are as follows.

Lemma 2. *Given the target function* $\psi = tr((\widetilde{\chi}\prod_k^K \times_k D_k)\prod_k^K \times_k D_k^T \widetilde{\chi}^T)$, $\widetilde{\chi} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$, $D_k \in \mathbb{R}^{J_k \times H_k}$, *each element in* $\{D_k\}$ *satisfies formula (12). Then, the partial derivative of the target function* ψ *to* D_{k_0} *where* $k_0 \in [1, K]$ *is* $\partial\psi/\partial D_{k_0}$, $\partial\psi/\partial D_{k_0} = \varepsilon(\widetilde{\chi}\prod_k^K \times {}_k D_k)\prod_k^K \times_k D_k^T$, *where* ε *is a constant.*

Proof 3. We use κ to represent $\tau\prod_k^K \times_k D_k$; τ is the identity tensor, $\tau \in \mathbb{R}^{J_1 \times J_2 \times \cdots J_K}$. Then, the target function ψ can be rewritten as $\psi = \text{tr}(\widetilde{\chi}\kappa\kappa^{\mathrm{T}}\widetilde{\chi}^{\mathrm{T}})$. We use ρ to represent $\kappa\kappa^{\mathrm{T}}$. We can obtain that $\partial\phi/\partial\widetilde{\chi} = \partial\text{tr}(\widetilde{\chi}\kappa\kappa^{\mathrm{T}}\widetilde{\chi}^{\mathrm{T}})/\partial\widetilde{\chi} = \partial\text{tr}(\widetilde{\chi}\rho\widetilde{\chi}^{\mathrm{T}})/\partial\widetilde{\chi}$. From the function derivation rule, we can get the following formula: $\partial\text{tr}(\widetilde{\chi}\kappa\kappa^{\mathrm{T}}\widetilde{\chi}^{\mathrm{T}})/\partial\widetilde{\chi} = \partial\text{tr}(\widetilde{\chi}\rho\widetilde{\chi}^{\mathrm{T}})/\partial\widetilde{\chi} = (\rho\widetilde{\chi}^{\mathrm{T}} + \rho^{\mathrm{T}}\widetilde{\chi}^{\mathrm{T}})^{\mathrm{T}}$; thus,

$$\frac{\partial\phi}{\partial\widetilde{\chi}} = 2\widetilde{\chi}\kappa\kappa^{\mathrm{T}} = \varepsilon\left(\widetilde{\chi}\prod_k^K \times_k D_k\right)\prod_k^K \times_k D_k^{\mathrm{T}}, \tag{19}$$

where ε is a constant, $\varepsilon = 2$.

After the aforementioned analysis, Proof 4 gives the solution to Problem 3 and its mathematical proof process while combining Lemma 2 and Proof 3. □

Proof 4. We use υ to represent $\chi\prod_k^K \times_k C_k$ and γ to represent $\tau\prod_k^K \times_k D_k$, where τ is the identity tensor, $\tau \in \mathbb{R}^{J_1 \times J_2 \times \cdots J_K}$. Then, the function F_{RTD} can be rewritten as $F_{\text{RTD}} = \|\upsilon - \widetilde{\chi}\gamma\|_{\mathrm{F}}^2$, that is $F_{\text{RTD}} = \text{tr}((\upsilon - \widetilde{\chi}\gamma)^{\mathrm{T}}(\upsilon - \widetilde{\chi}\gamma))$. Known by the definition of function tr, $\text{tr}(\gamma^{\mathrm{T}}\widetilde{\chi}^{\mathrm{T}}\upsilon) = \text{tr}(\widetilde{\chi}^{\mathrm{T}}\upsilon\gamma^{\mathrm{T}})$. Then, we can get the following equation: $\partial F_{\text{RTD}}/\partial\widetilde{\chi} = (\partial\text{tr}(\upsilon^{\mathrm{T}}\upsilon) - 2\partial\text{tr}(\widetilde{\chi}^{\mathrm{T}}\upsilon\gamma^{\mathrm{T}}) + \partial\text{tr}(\widetilde{\chi}\gamma\gamma^{\mathrm{T}}\widetilde{\chi}^{\mathrm{T}}))/\partial\widetilde{\chi}$. It can be derived from the function derivation rule and Lemma 2 that

$$\frac{\partial F_{\text{RTD}}}{\partial\widetilde{\chi}} = \varepsilon\left(\widetilde{\chi}\gamma\gamma^{\mathrm{T}} - \upsilon\gamma^{\mathrm{T}}\right), \tag{20}$$

where ε is a constant, $\varepsilon = 2$. Let $(\partial F_{\text{RTD}}/\partial\widetilde{\chi})$ be zero. We can get the final solution of Problem 3 by combining formula (12).

$$\widetilde{\chi} = \left(\chi\prod_k^K \times_k C_k\right)\prod_k^K \times_k D_k^{\mathrm{T}}. \tag{21}$$

Algorithm 2 implements RTD by using Algorithm 1. Function TSPA in line 1 represents the implementation of Algorithm 1, and the inputs are χ, η, and w_η. Function F_RTD in line 2 shows the calculation of $\widetilde{\chi}$ using equations (18)–(21). Finally, the core tensor of χ is obtained by using Algorithm 2, which approximates the restricted tensor η on the layer of tensor structure and elements information. □

4.2. rdRNN. This study proposes a new RNN called rdRNN. Unlike traditional RNN, rdRNN sets up a new gate based on the bidirectional RNN. The new gate uses the similarity matrix between samples as a parameter of the deep neural network training model. Compared with the original bidirectional RNN, the classification result of rdRNN is more accurate and stable. For the intelligent judgment of legal cases, the original deep neural network method does not consider the correlation between legal cases. This disregard may lead to bias in the final case classification. For example, the verdict of a legal case is inconsistent with the description

Input: Tensor χ which represents the original legal case data, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$, the restricted tensor η, $\eta \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$, and its weight w_η.
Output: The core tensor $\widetilde{\chi}$ which is close to the restricted tensor η in the layer of tensor structure and elements value, $\widetilde{\chi} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_K}$.
// Solving the factor matrix sets $\{C_k\}$ and $\{D_k\}$ using Algorithm 1
$\{C_k\}, \{D_k\} = \mathrm{TSPA}(\chi, \eta, w_\eta)$;
$\widetilde{\chi} = \mathrm{F\_RTD}(\chi, \{C_k\}, \{D_k\})$;
return $\widetilde{\chi}$;

Algorithm 2: The restricted tensor decomposition method.

of the case. To solve this problem, rdRNN fully considers the judgment results of legal cases that are similar to the case to be judged. rdRNN uses these results as a parameter of the deep neural network training model and realizes an efficient and accurate classification of multiple accusations in legal cases.

The following section shows the training of rdRNN's deep neural network while using Rmsprop as its optimization function:

(i) We use the dataset $\left\{\widetilde{\chi}^{(n)}, L^{(n)}\right\}$ and the restricted tensor η as inputs of rdRNN. $\widetilde{\chi}^{(n)}$ is the core tensor of $\chi^{(n)}$, which represents the legal case. $\widetilde{\chi}^{(n)}$ is obtained by the RTD algorithm with $\chi^{(n)}$ and η as its inputs. $L^{(n)}$ represents the category label of $\chi^{(n)}$ according to the judgment result of legal case.

(ii) In this study, we combine rdRNN with the softmax layer to complete the classification of legal cases. For sample $\chi^{(n)}$, assuming $h^{(n)}$ is the output vector of rdRNN, the softmax layer implements the mapping of $h^{(n)}$ to the legal case category $L^{(n)}$.

(iii) We use cross entropy croen_F as the loss function to update rdRNN. rdRNN uses its forward propagation algorithm and error backpropagation formulas to iterate over the values of parameters in neural networks, such as weight matrices $\{w_d\}$ and bias terms $\{b_d\}$ that are associated with relationship gate, and restricted tensor η, where d is the number of hidden layers.

(iv) We select Adam as the optimization function of rdRNN, and Rmsprop completes the optimization and calculation of parameters $\{w_d\}$, $\{b_d\}$, and η by using $\partial \mathrm{crosen\_F}/\partial w_d$, $\partial \mathrm{crosen\_F}/\partial b_d$, and $\partial \mathrm{crosen\_F}/\partial \eta$.

4.2.1. Calculation of Forward Propagation in rdRNN. In this study, we fully consider the relationship between legal cases and set up a new gate to complete the classification of legal cases, eliminate contingency errors as much as possible, and avoid inconsistencies between the predicted judgment result and the actual case. Relationship control gate r_t is used to control the similar relationship between legal cases. r_t helps the rdRNN deep neural network make an intelligent judgment by using the judgment results of cases that are similar to the case to be judged.

rdRNN can be divided into forward and backward LSTM. These networks do not have obvious differences, except for the opposite propagation direction. In the case of rdRNN forward LSTM propagation network, the formal description of relationship control gate r_t is as follows:

$$r_t = \sigma\left(w_r \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_r\right), \tag{22}$$

where w_r and b_r are the weight matrix and bias term of the relational control gate r_t, respectively, σ is the activation function, i.e., the sigmoid function, h_{t-1} is the output unit state of the neuron at time $t-1$, and x_t is the input value of the neuron at time t.

In the forward LSTM network, the output of each neuron at time t is calculated by the following formula:

$$\begin{aligned} \begin{bmatrix} r_t \\ f_t \\ i_t \\ o_t \\ \widetilde{c}_t \end{bmatrix} &= \begin{bmatrix} \sigma\left(\mathrm{net}_{r,t}\right) \\ \sigma\left(\mathrm{net}_{f,t}\right) \\ \sigma\left(\mathrm{net}_{i,t}\right) \\ \sigma\left(\mathrm{net}_{o,t}\right) \\ \tanh\left(\mathrm{net}_{\widetilde{c},t}\right) \end{bmatrix} \\ &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\begin{bmatrix} w_{rh}, w_{rx} \\ w_{fh}, w_{fx} \\ w_{ih}, w_{ix} \\ w_{oh}, w_{ox} \\ w_{ch}, w_{cx} \end{bmatrix} \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + \begin{bmatrix} b_r \\ b_f \\ b_i \\ b_o \\ b_c \end{bmatrix} \right), \end{aligned} \tag{23}$$

where r_t, f_t, i_t, and o_t are the relational control, forget, input, and output gates, respectively; $\widetilde{c}_t$ is the unit status of current inputs; $\mathrm{net}_{r,t}$, $\mathrm{net}_{f,t}$, $\mathrm{net}_{i,t}$, and $\mathrm{net}_{o,t}$ are the weighted inputs of their corresponding gates at time t; $\mathrm{net}_{c,t}$ is the weighted inputs of input state generation function tanh; σ is the activation function, i.e., the sigmoid function,

$\sigma(x) = 1/(1+e^{-x})$; tanh is the hyperbolic tangent function, $\tanh(x) = (e^{x}-e^{-x})/(e^{x}+e^{-x})$; w_r is the weight matrix of relationship control gate r_t, $w_r = [w_{rh}, w_{rx}]$; w_f, w_i, and w_o are expressed in the same manner as w_r; and b_r, b_f, b_i, and b_o are the bias terms of their corresponding activation functions.

Subsequently, the unit state of the current moment c_t is calculated by f_t, c_{t-1}, i_t, and $\tilde{c}_t$. The calculation formula is expressed as follows:

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t. \tag{24}$$

The final output of the forward LSTM neural network at time t is calculated by o_t, c_t, and r_t and the similar list of x. It is described as follows:

$$h_t = o_t * \tanh(c_t) + r_t * \left(\frac{1}{|\mathrm{List}(x)|} \sum_{x_0 \in \mathrm{List}(x)} \mathrm{Sim}(x_0, x) h_{x_0} \right), \tag{25}$$

where $\mathrm{List}(x)$ is composed of legal cases where the similarity with x is greater than a threshold Max_sim so far. h_{x_0} refers to the output of the forward LSTM neural network that corresponds to legal case x_0. $\mathrm{Sim}(\cdot)$ is a function that calculates the similarity between legal cases. In this study, we set function Sim as the weight of the Euclidean distance and the cosine distance between legal cases.

$$\mathrm{Sim}(x, x_0) = D_{\mathrm{Euclidean}}(x, x_0) + w_d D_{\mathrm{cosine}}(x, x_0), \tag{26}$$

where $D_{\mathrm{Euclidean}}$ and D_{cosine} refer to the Euclidean distance and the cosine distance between the vectors x and x_0, respectively. w_d is the weight matrix.

4.2.2. Calculation of Backpropagation in rdRNN. In this section, we describe in detail the backpropagation algorithm of the rdRNN neural network, including the backpropagation of the error along time and the hidden layer. In rdRNN, forward and backward LSTM neural networks have the same principle in the backpropagation algorithm. Therefore, this section mainly uses forward LSTM as an example.

Given the error term at time t δ_t, $\delta_t = (\partial \mathrm{crosen\_F}/\partial h_t)$. Calculation of the backpropagation algorithm of the error term along time is to calculate the value of $\delta_{t-1} = (\partial \mathrm{crosen\_F}/\partial h_{t-1})$. The full derivative formula shows that

$$\delta_{t-1} = \frac{\partial \mathrm{crosen\_F}}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}}. \tag{27}$$

Equations (23)–(25) show that r_t, f_t, i_t, o_t, and $\tilde{c}_t$ are all functions of h_{t-1}. Then, we can obtain

$$\begin{cases} \delta_{o,t} = \delta_t \dfrac{\partial h_t}{\partial o_t} \dfrac{\partial o_t}{\partial \mathrm{net}_{o,t}}, \\ \delta_{f,t} = \delta_t \dfrac{\partial h_t}{\partial c_t} \dfrac{\partial c_t}{\partial f_t} \dfrac{\partial f_t}{\partial \mathrm{net}_{f,t}}, \\ \delta_{i,t} = \delta_t \dfrac{\partial h_t}{\partial c_t} \dfrac{\partial c_t}{\partial i_t} \dfrac{\partial i_t}{\partial \mathrm{net}_{i,t}}, \\ \delta_{\tilde{c},t} = \delta_t \dfrac{\partial h_t}{\partial c_t} \dfrac{\partial c_t}{\partial \tilde{c}_t} \dfrac{\partial \tilde{c}_t}{\partial \mathrm{net}_{\tilde{c},t}}, \\ \delta_{r,t} = \delta_t \dfrac{\partial h_t}{\partial r_t} \dfrac{\partial r_t}{\partial \mathrm{net}_{r,t}}, \end{cases} \tag{28}$$

$$\begin{cases} \dfrac{\partial \mathrm{net}_{o,t}}{\partial h_{t-1}} = w_{oh}, \\ \dfrac{\partial \mathrm{net}_{f,t}}{\partial h_{t-1}} = w_{fh}, \\ \dfrac{\partial \mathrm{net}_{i,t}}{\partial h_{t-1}} = w_{ih}, \\ \dfrac{\partial \mathrm{net}_{\tilde{c},t}}{\partial h_{t-1}} = w_{ch}, \\ \dfrac{\partial \mathrm{net}_{r,t}}{\partial h_{t-1}} = w_{rh}. \end{cases}$$

The formula on the left represents the variable declaration, and the formula on the right is calculated from equation (23). According to equations (27) and (28), we can further derive the following formula:

$$\delta_{t-1} = \delta_{o,t} w_{oh} + \delta_{f,t} w_{fh} + \delta_{i,t} w_{ih} + \delta_{\tilde{c},t} w_{ch} + \delta_{r,t} w_{rh}, \tag{29}$$

where $\delta_{o,t} = \delta_t * \tanh(c_t) * o_t * (1-o_t)$, $\delta_{f,t} = \delta_t * o_t * (1-\tanh(c_t)^2) * c_{t-1} * f_t * (1-f_t)$, $\delta_{i,t} = \delta_t * o_t * (1-\tanh(c_t)^2) * \tilde{c}_t * i_t * (1-i_t)$, $\delta_{\tilde{c},t} = \delta_t * o_t * (1-\tanh(c_t)^2) * i_t * (1-\tilde{c}_t^2)$. According to relationship control gate r_t, equations (23) and (25), we determine that

$$\delta_{r,t} = \delta_t * \left(\frac{1}{|\mathrm{List}(x)|} \sum_{x_0 \in \mathrm{List}(x)} \mathrm{Sim}(x_0, x) h_{x_0} \right) * r_t * (1-r_t). \tag{30}$$

From equation (29), we can finally figure out the calculation method of the error term in rdRNN is passed from the current moment t to any time k.

Then, we describe in detail the transmission of error between the hidden layers of rdRNN. The error term of the lth hidden layer in rdRNN is assumed to be the partial derivative of the error function crosen_F versus the weighted input net^l. In rdRNN, the input of the lth hidden layer at time t is x_t^l.

$$x_t^l = \text{active}F^{l-1}\left(\text{net}_t^{l-1}\right), \tag{31}$$

where $\text{active}F^{l-1}$ denotes the activation function of the $l-1$th hidden layer in rdRNN and net_t^{l-1} denotes the weighted input of the $l-1$th hidden layer at time t.Given the error term of the lth hidden layer at time t δ_t^l, $\delta_t^l = (\partial\text{crosen\_F}/\partial\text{net}_t^l)$, the calculation of error propagation between hidden layers is to figure out the value of δ_t^{l-1}, where

$$\delta_t^{l-1} = \frac{\partial\text{crosen\_F}}{\partial\text{net}_t^{l-1}}. \tag{32}$$

According to equations (23), (31), and (32), $\text{net}_{r,t}^l$, $\text{net}_{f,t}^l$, $\text{net}_{i,t}^l$, $\text{net}_{o,t}^l$, and $\text{net}_{\tilde{c},t}^l$ are all functions of x_t^l, and x_t^l is a function of net_t^{l-1}. Therefore, the full derivative formula shows that

$$\begin{aligned}\delta_t^{l-1} = {} & \frac{\partial\text{crosen\_F}}{\partial\text{net}_{r,t}^l}\frac{\partial\text{net}_{r,t}^l}{\partial x_t^l}\frac{\partial x_t^l}{\partial\text{net}_t^{l-1}} + \frac{\partial\text{crosen\_F}}{\partial\text{net}_{f,t}^l}\frac{\partial\text{net}_{f,t}^l}{\partial x_t^l}\frac{\partial x_t^l}{\partial\text{net}_t^{l-1}} \\ & + \frac{\partial\text{crosen\_F}}{\partial\text{net}_{i,t}^l}\frac{\partial\text{net}_{i,t}^l}{\partial x_t^l}\frac{\partial x_t^l}{\partial\text{net}_t^{l-1}} + \frac{\partial\text{crosen\_F}}{\partial\text{net}_{o,t}^l}\frac{\partial\text{net}_{o,t}^l}{\partial x_t^l}\frac{\partial x_t^l}{\partial\text{net}_t^{l-1}} \\ & + \frac{\partial\text{crosen\_F}}{\partial\text{net}_{\tilde{c},t}^l}\frac{\partial\text{net}_{\tilde{c},t}^l}{\partial x_t^l}\frac{\partial x_t^l}{\partial\text{net}_t^{l-1}}.\end{aligned} \tag{33}$$

The following formula can be obtained by further calculation:

$$\begin{aligned}\delta_t^{l-1} = {} & \left(\delta_{r,t}^l w_{rx} + \delta_{f,t}^l w_{fx} + \delta_{i,t}^l w_{ix} + \delta_{o,t}^l w_{ox} + \delta_{\tilde{c},t}^l w_{cx}\right) \\ & * \text{gra}\left(\text{active}F^{l-1}\left(\text{net}_t^{l-1}\right)\right),\end{aligned} \tag{34}$$

where $\text{gra}\left(\text{active}F^{l-1}\left(\text{net}_t^{l-1}\right)\right)$ is the derivative of function $\text{active}F^{l-1}$ at net_t^{l-1}.

According to equations (27)–(34), we can derive the partial derivative of the loss function crosen_F to the weight matrix set $\{w_{\ d}\}$ and the bias term set $\{b_{\ d}\}$ in rdRNN. Given that $\partial\text{crosen\_F}/\partial w_{rh} = \sum_{j=1}^t \partial\text{crosen\_F}/\partial w_{rh,j} = \sum_{j=1}^t (\partial\text{crosen\_F}/\partial\text{net}_{r,j})(\partial\text{net}_{r,j}/\partial w_{rh,j})$, we obtain

$$\begin{cases}\dfrac{\partial\text{crosen\_F}}{\partial w_{rx}} = \displaystyle\sum_{j=1}^t \delta_{r,j} h_{j-1}, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{fx}} = \displaystyle\sum_{j=1}^t \delta_{f,j} h_{j-1}, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{ix}} = \displaystyle\sum_{j=1}^t \delta_{i,j} h_{j-1}, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{ox}} = \displaystyle\sum_{j=1}^t \delta_{o,j} h_{j-1}, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{cx}} = \displaystyle\sum_{j=1}^t \delta_{\tilde{c},j} h_{j-1},\end{cases}$$

$$\begin{cases}\dfrac{\partial\text{crosen\_F}}{\partial b_r} = \displaystyle\sum_{j=1}^t \delta_{r,j}, \\ \dfrac{\partial\text{crosen\_F}}{\partial b_f} = \displaystyle\sum_{j=1}^t \delta_{f,j}, \\ \dfrac{\partial\text{crosen\_F}}{\partial b_i} = \displaystyle\sum_{j=1}^t \delta_{i,j}, \\ \dfrac{\partial\text{crosen\_F}}{\partial b_o} = \displaystyle\sum_{j=1}^t \delta_{o,j}, \\ \dfrac{\partial\text{crosen\_F}}{\partial b_c} = \displaystyle\sum_{j=1}^t \delta_{\tilde{c},j},\end{cases} \tag{35}$$

$$\begin{cases}\dfrac{\partial\text{crosen\_F}}{\partial w_{rx}} = \delta_{r,j} x_t, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{fx}} = \delta_{f,j} x_t, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{ix}} = \delta_{i,j} x_t, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{ox}} = \delta_{o,j} x_t, \\ \dfrac{\partial\text{crosen\_F}}{\partial w_{cx}} = \delta_{\tilde{c},j} x_t.\end{cases}$$

4.2.3. Calculation of the Partial Derivative of Loss Function crosen_F to Restricted Tensor η. This study proposes a new intelligent method for judging legal cases called RnRTD, which combines rdRNN and RTD to complete the classification of legal cases. In the process of training the RnRTD neural network, a new problem is involved: updating of the value of the restricted tensor η so that it can continuously approximate the tensor value that is most beneficial for improving the classification accuracy of the RnRTD algorithm.

The crux to solving this problem is to calculate the partial derivative of the loss function crosen_F to the restricted tensor η, that is, ∂crosen_F/$\partial\eta$. Directly solving the value of ∂crosen_F/$\partial\eta$ is difficult. We can use the full derivative rule to obtain that

$$\frac{\partial \text{crosen\_F}}{\partial \eta} = \sum_{n}^{N} \frac{\partial \text{crosen\_F}}{\partial \widetilde{\chi}^{(n)}} \frac{\partial \widetilde{\chi}^{(n)}}{\partial \eta}. \tag{36}$$

The backward propagation formula of rdRNN shows that $\partial\text{crosen\_F}/\partial\widetilde{\chi}^{(n)} = (\partial\text{crosen\_F}/\partial\text{net}_1)(\partial\text{net}_1/\ \partial\widetilde{\chi}^{(n)}) = \delta_1 w_0$. According to equations (19)–(21), we determine that $\partial\widetilde{\chi}^{(n)}/\partial\eta = \partial[(\chi^{(n)}\prod_k^K \times_k C_k^{(n)})\prod_{k_1}^K \times_{k_1} U_{k_1}^{(n)}\text{T}]/\partial\eta$. From equations (14)–(17), we know that $C_k^{(n)}$ and $D_k^{(n)}$ are all functions of η. Therefore, the function full derivative rule shows that $\partial\widetilde{\chi}^{(n)}/\partial\eta = \sum_k^K (\partial\widetilde{\chi}^{(n)}/\partial C_k^{(n)})(\partial C_k^{(n)}/\partial\eta) + \sum_k^K (\partial\widetilde{\chi}^{(n)}/\partial D_k^{(n)^{\text{T}}})(\partial D_k^{(n)}\text{T}/\partial\eta)$, that is

$$\frac{\partial \widetilde{\chi}^{(n)}}{\partial \eta} = \sum_{k}^{K} \text{cal}C\left(C_k^{(n)}\right) + \sum_{k}^{K} \text{cal}D\left(D_k^{(n)}\text{T}\right), \tag{37}$$

while

$$\begin{cases} \text{cal}C\left(C_k^{(n)}\right) = \chi^{(n)} \prod_{k_1 \neq k}^{K} \times_{k_1} C_{k_1}^{(n)} \times_k \dfrac{\partial C_k^{(n)}}{\partial \eta} \prod_{k_2}^{K} \times_{k_2} D_{K_2}^{(n)}\text{T}, \\ \text{cal}D\left(D_k^{(n)}\text{T}\right) = \chi^{(n)} \prod_{k_1}^{K} \times_{k_1} C_{k_1}^{(n)} \times_k \dfrac{\partial D_k^{(n)}\text{T}}{\partial \eta} \prod_{k_2 \neq k}^{K} \times_{k_2} D_{K_2}^{(n)}\text{T}. \end{cases} \tag{38}$$

Algorithm 3 provides the optimization process of RnRTD proposed in this study. max_iter in line 2 represents the total number of training sessions of RnRTD. batch_size in line 6 represents the number of samples per batch while training RnRTD. The RTD tensor decomposition method in line 7 corresponds to Algorithm 2. rdRNN_forwardprop in line 8 and rdRNN_backprop in line 9 represent forward propagation and error backpropagation algorithms for rdRNN, respectively, which are the implementations of Sections 4.2.1–4.2.3. In line 11, we use the Adam algorithm to realize parameter optimization of RnRTD neural network.

4.2.4. Loss Function crosen_F and Softmax Layer. In Algorithm 3, we use the softmax function softmax to calculate the probability that $\chi^{(n)}$ belongs to each type of legal case according to judgment results.

Definition 9. Given a set of samples of legal cases and their corresponding outputs of RnRTD $\{(\chi^{(n)}, S^{(n)})\}$, the probability that $\chi^{(n)}$ belongs to each type of legal case is calculated by

$$L_{1q}^{(n)} = \text{softmax}\left(S^{(n)}\right) = \frac{e^{S_q^{(n)}}}{\sum_r^Q e^{S_r^{(n)}}}, \tag{39}$$

where $L_{1q}^{(n)}$ represents the qth element of $L_1^{(n)}$.

In this study, cross entropy is used as the loss function crosen_F to calculate the error of RnRTD. We define crosen_F as follows:

Definition 10. A set of samples of legal cases and their corresponding legal case types $\{(\chi^{(n)}, L^{(n)})\}$ is given. The predicted legal case category of $\chi^{(n)}$ is $L_1^{(n)}$, which is calculated by RnRTD, and then

$$\text{crosen\_F}\left(\left\{L^{(n)}\right\}, \left\{L_1^{(n)}\right\}\right) = \sum_{n}^{N} \sum_{q}^{Q} L_q^{(n)} \log L_{1q}^{(n)}, \tag{40}$$

where N represents the number of samples of legal cases and q represents the dimension of $L^{(n)}$ and $L_1^{(n)}$, that is, the number of types of legal cases.

5. Results

5.1. Description of Experimental Data. We use nearly 1.8 million historical legal cases obtained from a Chinese refereeing study network. These legal cases involve more than 200 types of accusations, including theft, intentional assault, smuggling, fraud, and deliberate destruction of public property. Approximately 400,000 cases involve theft, and about 200,000 cases involve intentional assault. The number of accusations involved in each case ranges from 1 to 23.

Figure 8(a) shows the distribution of various accusations in the legal case data used in this study. The abscissa indicates the accusation index. For example, index 1 corresponds to bribery, and index 2 corresponds to rape. The ordinate indicates the proportion of cases involving the accusation that occupy the overall cases. Figure 8(a) shows that the number of cases involving theft is the highest in the database used in this article.

Figure 8(b) shows the distribution of the number of accusations involved in each case. The abscissa indicates the number of accusations involved in the case, and the ordinate indicates the proportion of cases in the corresponding number of accusations. Figure 8(b) shows that the highest number of accusations involves three cases.

5.2. Baseline Approaches. Given that multiaccusation judgment based on deep neural network and tensor decomposition is rarely studied, according to the limited tensor decomposition method RTD and the relation-based recurrent neural network rdRNN, we use the following method for a comparison with RnRTD proposed in this study:

Input: $\{(\chi^{(n)}, L^{(n)})\}$, where $\chi^{(n)}$ represents the legal cases and $L^{(n)}$ represents the category of legal case corresponding to $\chi^{(n)}$ according to judgment results. The size of η, $\{w_r\}$, $\{w_f\}$, $\{w_i\}$, $\{w_o\}$, $\{w_c\}$, $\{b_r\}$, $\{b_f\}$, $\{b_i\}$, $\{b_o\}$ and $\{b_c\}$, where $w_r = [w_{rh}, w_{rx}]$, $w_f = [w_{fh}, w_{fx}]$, $w_i = [w_{ih}, w_{ix}]$, $w_o = [w_{oh}, w_{ox}]$ and $w_c = [w_{ch}, w_{cx}]$
Output: The optimal restricted tensor η, parameters of rdRNN $\{w_r\}$, $\{w_f\}$, $\{w_i\}$, $\{w_o\}$, $\{w_c\}$ and $\{b_r\}$, $\{b_f\}$, $\{b_i\}$, $\{b_o\}$, $\{b_c\}$.
Initialize the restricted tensor η, parameters of rdRNN $\{w_r\}$, $\{w_f\}$, $\{w_i\}$, $\{w_o\}$, $\{w_c\}$ and $\{b_r\}$, $\{b_f\}$, $\{b_i\}$, $\{b_o\}$, $\{b_c\}$
for iter = 1 to max_iter **do**
  Set batch_indices to zero;
  while batch_indices ≤ dataset_size **do**
    Set $\{\partial\text{crosen\_F}/\partial\eta\}$, $\{\partial\text{crosen\_F}/\partial w_r\}$, $\{\partial\text{crosen\_F}/\partial w_f\}$, $\{\partial\text{crosen\_F}/\partial w_i\}$, $\{\partial\text{crosen\_F}/\partial w_o\}$, $\{\partial\text{crosen\_F}/\partial w_c\}$ and $\{\partial\text{crosen\_F}/\partial b_r\}$, $\{\partial\text{crosen\_F}/\partial b_f\}$, $\{\partial\text{crosen\_F}/\partial b_i\}$, $\{\partial\text{crosen\_F}/\partial b_o\}$, $\{\partial\text{crosen\_F}/\partial b_c\}$ to zero;
    for i = batch_indices to (batch_indices + batch_size) **do**
      $\widetilde{\chi}^{(i)} = \text{RTD}(\chi^{(i)}, \eta)$;
      $L_1^{(i)} = \text{rdRNN\_forwardprop}(\widetilde{\chi}^{(i)}, \{w_r\}, \{w_f\}, \{w_i\}, \{w_o\}, \{w_c\}, \{b_r\}, \{b_f\}, \{b_i\}, \{b_o\}, \{b_c\})$;
      $\{\partial\text{crosen\_F}/\partial\eta\}$, $\{\partial\text{crosen\_F}/\partial w_r\}$, $\{\partial\text{crosen\_F}/\partial w_f\}$, $\{\partial\text{crosen\_F}/\partial w_i\}$, $\{\partial\text{crosen\_F}/\partial w_o\}$, $\{\partial\text{crosen\_F}/\partial w_c\}$, $\{\partial\text{crosen\_F}/\partial b_r\}$, $\{\partial\text{crosen\_F}/\partial b_f\}$, $\{\partial\text{crosen\_F}/\partial b_i\}$, $\{\partial\text{crosen\_F}/\partial b_o\}$, $\{\partial\text{crosen\_F}/\partial b_c\}$+ = $\text{rdRNN\_backprop}(\widetilde{\chi}^{(i)}, L_1^{(i)}, L^{(i)}, \eta, \{w_r\}, \{w_f\}, \{w_i\}, \{w_o\}, \{w_c\}, \{b_r\}, \{b_f\}, \{b_i\}, \{b_o\}, \{b_c\})$;
    end
  $\eta, \{w_r\}, \{w_f\}, \{w_i\}, \{w_o\}, \{w_c\}, \{b_r\}, \{b_f\}, \{b_i\}, \{b_o\}, \{b_c\} = \text{Adam}(\eta, \{w_r\}, \{w_f\}, \{w_i\}, \{w_o\}, \{w_c\}, \{b_r\}, \{b_f\}, \{b_i\}, \{b_o\}, \{b_c\}, \{\partial\text{crosen\_F}/\partial\eta\}, \{\partial\text{crosen\_F}/\partial w_r\}, \{\partial\text{crosen\_F}/\partial w_f\}, \{\partial\text{crosen\_F}/\partial w_i\}, \{\partial\text{crosen\_F}/\partial w_o\}, \{\partial\text{crosen\_F}/\partial w_c\}, \{\partial\text{crosen\_F}/\partial b_r\}, \{\partial\text{crosen\_F}/\partial b_f\}, \{\partial\text{crosen\_F}/\partial b_i\}, \{\partial\text{crosen\_F}/\partial b_o\}, \{\partial\text{crosen\_F}/\partial b_c\})$;
  end
end
return η, $\{w_r\}$, $\{w_f\}$, $\{w_i\}$, $\{w_o\}$, $\{w_c\}$ and $\{b_r\}$, $\{b_f\}$, $\{b_i\}$, $\{b_o\}$, $\{b_c\}$;

Algorithm 3: The framework of RnRTD algorithm.

(i) This study uses a series of deep neural networks based on convolutional or cyclic neural networks, including TextCNN, TextCNN attention, TextRNN, TextRNN attention, LSTM, and Bi-LSTM, as comparison methods for RnRTD.

(ii) This study uses deep neural networks with only the RTD tensor decomposition layer as comparison methods for RnRTD. Through experiments, we can derive the contribution of the RTD tensor decomposition layer to all deep neural networks.

(iii) This study uses neural networks with only the rdRNN method as comparison methods for RnRTD. Through these comparison experiments, we can derive the contribution of the relationship-based rdRNN strategy to all RNNs.

5.3. Data Preprocessing. In this study, the data preprocessing operation can be divided into two parts, namely, the modular representation of legal cases and the construction of the original tensor. Our legal case data preprocessing process can be described as follows:

(i) We organize each case in the legal case database into our preestablished case model, which divides the original case file into the defendant's statement, the plaintiff's statement, the content of the public prosecutions allegations, and the court's judgment.

(ii) We filter and clean the contents of each module in the legal cases, extract the words that are meaningful to our multiaccusation judgment method, and filter out redundant words, stop words, noisy words, and modal particles.

(iii) We train the word-to-vector model to obtain the word vector of the aforementioned vocabulary. Then, we obtain a matrix representation of each case module and derive the tensor representation of the entire legal case.

For Step (1), each case module may be spread across different paragraphs, and cases in different regions have different case descriptions. We extract and integrate them separately to arrive at a modular representation of the cases based on the description rules of case documents in each region. For Step (2), the extraction and filtering of the vocabulary in legal cases often requires professional legal background knowledge; otherwise, error filtering will occur. We filter words in legal case modules by using the legal professional vocabulary and the stop word list. For Step (3), word vectors are the basis for the accuracy of the entire deep neural network method. We use a number of Chinese corpus, such as corpus on Baidu Encyclopedia, Zhihu Questions and Answers, Sohu News, and Sina Weibo, to train the word-to-vector model.

The tensor representation of legal cases and the subsequent deep neural network classification method require each case to have the same number of words, and the number of words in 95% of the cases is below 300. Therefore, we perform a padding operation for cases where the number of words is less than 300. For cases with a vocabulary number greater than 300, we use the TF-IDF weight of the vocabulary to tailor the case vocabulary.

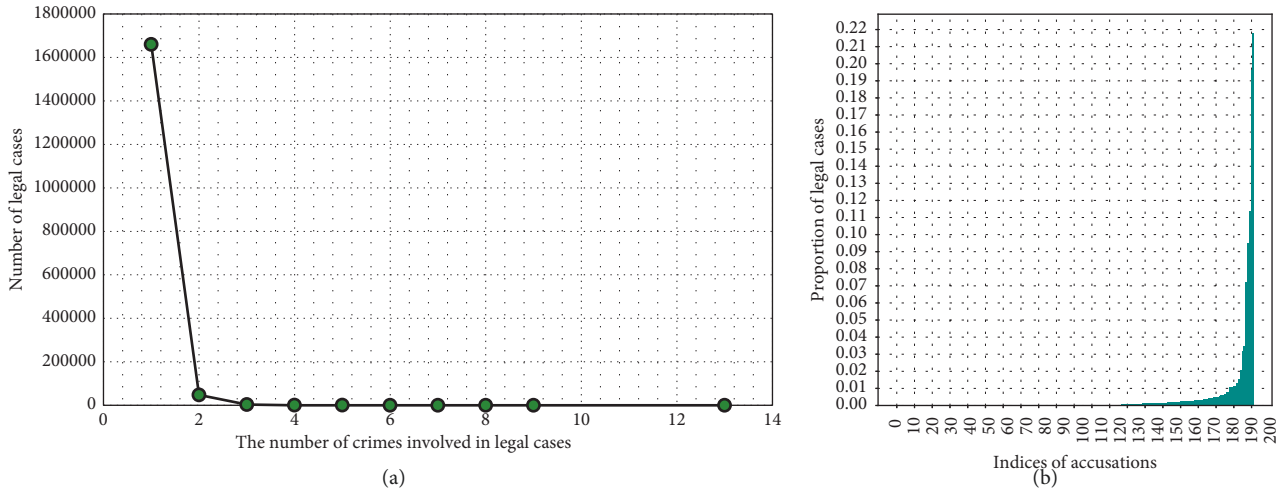

Figure 8: Statistics on allegations in legal cases.

5.4. Experimental Hyperparameter Setting. This section describes the hyperparameter settings involved in our proposed method. These settings include the restricted tensor η and weight matrix W_η in RTD and the size of the similar case list in the rdRNN method (i.e., the size of list $|*|$ in equation (25)).

The setting of restricted tensor η directly affects the convergence speed and accuracy of RnRTD. Our experiments show that a large rank of restricted tensor η corresponds to a high accuracy of the subsequent deep neural network algorithms. Conversely, a strong linear relationship between column or row vectors in η results in a low accuracy of the subsequent classification algorithms.

In this study, weight matrix W_η is used to scale the elements of the last mode in the original tensors. W_η adjusts the weights of certain words in the legal cases. For different accusations, the same vocabulary may have different weights in different types of cases. For example, derailment occupies a large and small weight in cases that involve bigamy and smuggling, respectively.

The size of the similar case list in rdRNN is an important indicator that determines the impact of the relationship between cases on the final classification result. If the length of the similar case list is set too long, then it is equivalent to strengthening the weak similarity between cases and weakening the strong similarity between cases. Furthermore, if the length of the similar case list is set too short, then it is equivalent to weakening the weak similarity between cases and strengthening the strong similarity between cases. After many experiments, we set the case similar list length to 50.

5.5. Experimental Results and Analysis. This section shows the superiority of the proposed RnRTD method for multiple accusations in legal cases relative to the baseline listed in Section 5.2 and provides the corresponding analysis.

Figure 9 shows a series of experimental results based on Bi-LSTM. The abscissa indicates the number of batch iterations, and the ordinate indicates the accuracy of the multiaccusation judgment methods in legal cases. In contrast with the original Bi-LSTM method and Bi-LSTM with only the rdRNN layer, Bi-LSTM with only RTD achieves stable accuracy at the highest speed as the number of batches increases.

The characteristics of RTD are important factors in the aforementioned phenomenon. On the basis of the restricted tensor η, RTD extracts the tensor elements and structure information that are most relevant to the multiaccusation judgment of legal cases from the original tensor. The weight of the vocabulary unrelated to a particular accusation is considerably weakened, and the weight of the vocabulary associated with a particular accusation is strengthened. The tensor dimension is greatly reduced, and the influence of irrelevant vocabulary on the classification algorithm is reduced. Subsequent neural network algorithms continuously iterate and optimize the restricted tensor and continuously adjust and correct the element values of the core tensor. RTD optimizes the original deep neural network algorithm from the lexical level.

In Figure 9, as the number of batches increases, the accuracy of Bi-LSTM with only the rdRNN layer becomes ultimately higher than that of Bi-LSTM with only the RTD layer. The reason is that rdRNN fully considers the similarity between different cases and has better discrimination for similar cases expressed by different language description methods. By setting the appropriate similar case list size, rdRNN fully considers cases that are similar to the current case and weighs their corresponding output states according to their similarity. rdRNN corrects and optimizes the original deep neural network from the case level.

Figure 10 shows the experimental results of the TextRNN-based RnRTD method. Similar to what is shown in Figure 9, TextRNN with only the RTD layer has the highest convergence speed as the number of batches increases compared with the original TextRNN and TextRNN with only the rdRNN layer.

The accuracy of the deep neural network method with only the rdRNN layer is not always higher than that with

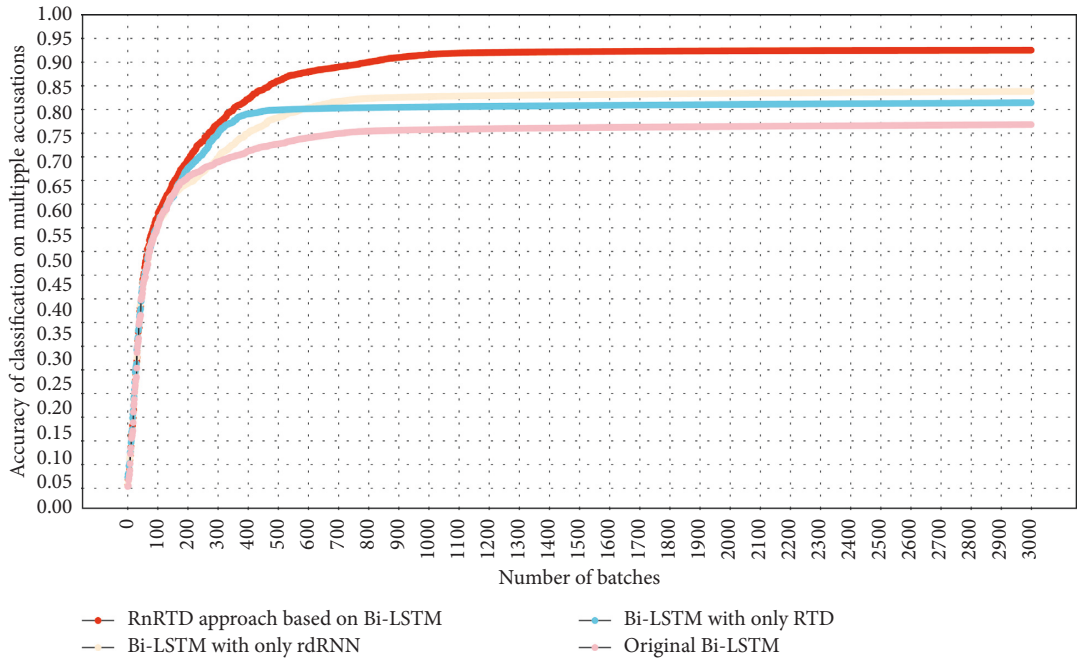

Figure 9: Experimental results of RnRTD based on Bi-LSTM.

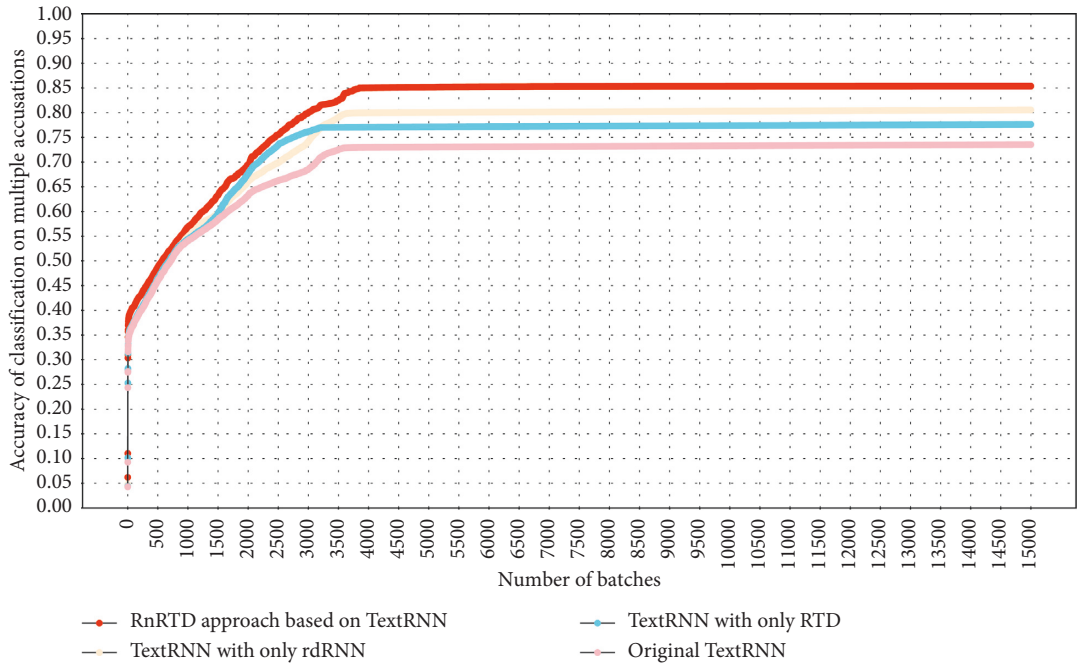

Figure 10: Experimental results of RnRTD based on TextRNN.

only the RTD layer. Although the rdRNN layer implements the correction and optimization of subsequent classification algorithms at the case level through the setting of similar case lists, the RTD layer also optimizes classification algorithms at the vocabulary level by setting the restricted tensor. Both methods achieve the final accuracy optimization but have different effects for various contexts. RnRTD combines the advantages of RTD and rdRNN to achieve rapid convergence and high classification accuracy.

Table 2 provides an experimental comparison of RnRTD methods based on multiple deep neural networks. RnRTD remarkably improves the classification accuracy of original neural networks for the classification of multiple accusations in legal cases. RTD and rdRNN layers also have considerable optimization effects on the original neural networks. RTD is applicable to all deep neural networks and can extract the main information carried by the data at the input layer to realize dimension reduction. rdRNN is an optimization

Table 2: Performance of RnRTD methods based on multiple deep neural networks.

Optimization method	Basic algorithm						
	TextCNN	TextRNN	TextCNN attention	TextRNN attention	LSTM	Bi-LSTM	GRU
Original method	0.70	0.73	0.74	0.76	0.75	0.77	**0.79**
Only with RTD	0.73	0.77	0.79	0.82	0.80	0.81	**0.83**
Only with rdRNN	0.74	0.80	0.82	0.83	0.83	**0.84**	0.82
With RnRTD	**0.80**	**0.85**	**0.85**	**0.90**	**0.91**	**0.93**	**0.91**

strategy that is suitable for RNNs. It fully considers the similarity between cases within a certain period and optimizes algorithms at the case level. For algorithms based on convolutional neural networks, we remove the relational control gate in rdRNN while retaining the similar case list. Then, the optimization of these algorithms is realized by the rdRNN layer.

The convolutional neural network is less effective than RNN because legal case data are time series data. In addition, the attention layer only changes the encoding of the input and does not change the structure of neural networks. For the problem of judgment for multiple accusations in legal cases, the attention layer is still difficult to compensate for due to the lack of timing information of TextCNN and the gradient disappearance and gradient explosion of the TextRNN algorithm. From the perspective of the rdRNN layer, GRU has fewer adjustable parameters than LSTM and Bi-LSTM, and optimization on the restricted tensor is relatively limited. Therefore, the Bi-LSTM neural network with RnRTD performs better than other neural network algorithms.

6. Discussion

In this study, we propose a new method for multiaccusation judgment in legal cases called RnRTD. RnRTD is a multilabel classification method based on tensor decomposition and RNNs. RnRTD consists of the tensor decomposition method with constraints and relation-driven RNN.

We propose a tensor decomposition method with constraints, namely, RTD. We use this method to extract the tensor structure and element information that are most favorable to the subsequent classification algorithm from the original tensor that represents the legal case. RTD continuously corrects and optimizes the values of elements in the core tensor through the weight matrix and restricted tensor; hence, it continues to improve the classification accuracy of the neural network. RTD optimizes neural network classification algorithms at the lexical level. We also propose a relation-driven RNN strategy called rdRNN. Unlike traditional recurrent and LSTM neural networks, rdRNN sets up a new gating switch, that is, the similarity list window. It controls the impact of cases similar to the current case on the output status of the current case. rdRNN optimizes neural network classification algorithms at the case level.

According to our experimental results, the RTD layer and the relation-driven cyclic neural network rdRNN have remarkable optimization effects on various deep neural network algorithms. However, no obvious relationship exists between the two. RTD and rdRNN have their own advantages in different contexts. In Figures 9 and 10, the accuracy of rdRNN is higher than that of RTD. The accuracy of rdRNN is not always higher than that of RTD. RTD achieves stable accuracy the fastest as the number of batches increases in both figures. The reason is the principal component extraction and dimensionality reduction of RTD itself.

RTD is suitable for almost all deep neural networks. It performs principal component extraction and dimensionality reduction on the original data at the input layer. It is similar to traditional principal component analysis methods, such as PCA [20] and SVD [21]. Several decomposition methods [22], such as Tucker and CP [23], are currently used for high-dimensional data. These methods extract the main elements and structural information of the matrix or tensor at the logical level according to the linear relationship of the elements in the matrix or tensor. However, the resulting new matrix or tensor structure is often unexplained. According to the traditional matrix or tensor decomposition method, supervising the completion of the principal component extraction work is difficult.

The proposed restricted tensor provides interpretability for the tensor decomposition operation. Under the influence of the restricted condition tensor, RTD retains the information in the original tensor that is beneficial to the subsequent neural network and removes useless information. For the overall classification algorithm, RTD reduces the weight of weak correlation information and improves the influence of strong correlation information on the classification model. In addition, the subsequent deep neural network algorithm will continuously update and optimize the constraints in RTD to guide the core tensor to retain information that is conducive for the classification model. RTD optimizes the classification model at the vocabulary level by combining the weight matrix with the restricted tensor.

rdRNN fully considers the similarities between cases and uses it as a factor that influences the output status of the current case. rdRNN optimizes the entire classification model at the case level. Generally, different regions may use different legal case description vocabularies, and rdRNN sets the output status of similar historical cases within a certain period as the reference value of the current case output state by setting a similar case window. Moreover, it sets the weight according to the similarity. RnRTD combines RTD and rdRNN to optimize the classification results from the perspective of case and vocabulary. When we use rdRNN to optimize algorithms based on the convolutional neural network, we remove the relationship control gate, retain only the similar case list in rdRNN, and realize the optimization operation of the rdRNN layer on the neural network.

7. Conclusion

In this study, we propose a new method for judging multiple crimes in legal cases, namely RnRTD. RnRTD consists of RTD and rdRNN. RTD is a tensor decomposition method with constraints. RTD decomposes the original tensor that represents a legal case into a core tensor under the guidance of restricted tensor. The resulting core tensor represents the main tensor structure and element information that is most favorable for improving the accuracy of subsequent classification algorithm. We propose the rdRNN algorithm and train it using obtained core tensors. rdRNN guides the tensor decomposition process in RTD by continuously optimizing the restricted tensor and finally makes RTD develop in the direction that is most beneficial to improve the classification accuracy of rdRNN.

Nevertheless, this study has several problems. For example, even with the RTD tensor decomposition layer, algorithms based on RNNs usually run very slowly. In our future work, we will attempt to reduce the computational complexity of the algorithm and increase its speed.

Data Availability

The legal cases data after processing used to support the findings of this study are currently under embargo while the research findings are commercialized. Requests for data, 6/12 months after publication of this article, will be considered by the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

In the research and writing work of this thesis, we would like to thank the teachers and students of the lab team (Network and Information Security Research Center, Harbin Institute of Technology) for their selfless help. This work was supported by the National Key Research and Development Program of China (2018YFC0830602 and 2016QY03D0501).

References

[1] U. Hahn and N. Chater, *Understanding Similarity: A Joint Project for Psychology, Case-Based Reasoning, and Law*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1998.

[2] V. Aleven, "Using background knowledge in case-based legal reasoning: a computational model and an intelligent learning environment," *Artificial Intelligence*, vol. 150, no. 1-2, pp. 183–237, 2003.

[3] T. Bench-Capon and G. Sartor, "A model of legal reasoning with cases incorporating theories and values," *Artificial Intelligence*, vol. 150, no. 1-2, pp. 97–143, 2003.

[4] A. S. A. Garcez, L. C. Lamb and D. M. Gabbay, *Neural-Symbolic Cognitive Reasoning*, Springer, Berlin, Germany, 2009.

[5] T. R. Besold, A. S. A. Garcez, S. Bader et al., "Neural-symbolic learning and reasoning: a survey and interpretation," 2017, https://arxiv.org/abs/1711.03902.

[6] A. A. Garcez, T. Besold, L. de Raedt et al., "Neural-symbolic learning and reasoning: contributions and challenges," in *Proceedings of the 2015 AAAI Spring Symposium Series*, Stanford, CA, USA, March 2015.

[7] A. S. A. Garcez, D. M. Gabbay, and L. C. Lamb, "A neural cognitive model of argumentation with application to legal inference and decision making," *Journal of Applied Logic*, vol. 12, no. 2, pp. 109–127, 2014.

[8] M. G. Venkateshmurthy, T. V. Geetha, and R. K. Subramanian, "Extraction and representation of facts from legal briefs," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 2, pp. 167–182, 1990.

[9] T. D. Breaux, A. I. Antn, and E. H. Spafford, "A distributed requirements management framework for legal compliance and accountability," *Computers and Security*, vol. 28, no. 1-2, pp. 8–17, 2009.

[10] D. A. Waterman, J. Paul, and M. Peterson, "Expert systems for legal decision making," *Expert Systems*, vol. 3, no. 4, pp. 212–226, 1986.

[11] R. Bartolini, A. Lenci, S. Montemagni, V. Pirrelli, and C. Soria, *Automatic Classification and Analysis of Provisions in Italian Legal Texts: A Case Study*, in *Proceedings of the On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops*, Agia Napa, Cyprus, October 2004.

[12] S. Joshi, P. M. Deshpande, and T. Hampp, "Improving the efficiency of legal e-discovery services using text mining techniques," in *Proceedings of the Srii Global Conference*, San Jose, CA, USA, April 2011.

[13] O.-M. Sulea, M. Zampieri, S. Malmasi, M. Vela, L. P. Dinu, and J. van Genabith, "Exploring the use of text classification in the legal domain," 2017, https://arxiv.org/abs/1710.09306.

[14] S. Bruninghaus and K. D. Ashley, "Progress in textual case-based reasoning: predicting the outcome of legal cases from text," in *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, USA, 2006.

[15] M. Welling and M. Weber, "Positive tensor factorization," *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1255–1261, 2001.

[16] R. J. Williams and D. Zipser, *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*, MIT Press, Cambridge, MA, USA, 1989.

[17] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 2002.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *Proceedings of the International Conference on Artificial Neural Networks ICANN*, pp. 850–855, Madrid, Spain, August 2002.

[19] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[20] E. Oja, "The nonlinear pca learning rule in independent component analysis," *Neurocomputing*, vol. 17, no. 1, pp. 25–45, 1997.

[21] M. Vozalis and K. Margaritis, "Using svd and demographic data for the enhancement of generalized collaborative filtering," *Information Sciences*, vol. 177, no. 15, pp. 3017–3037, 2007.

[22] A. Cichocki, R. Zdunek, and S. I Amari, "Nonnegative matrix and tensor factorization," *IEEE Signal Processing Magazine*, vol. 25, no. 3, pp. 54–65, 2009.

[23] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.