International Journal of Computational Intelligence Systems (2022) 15:48
https://doi.org/10.1007/s44196-022-00101-9

RESEARCH ARTICLE

An Improved AdaBoost for Prosecutorial Case-Workload Estimation via Case Grouping

Xin Min[1] · Wei Li[2] · Jinzhao Yang[1] · Weidong Xie[1] · Dazhe Zhao[1]

Received: 23 March 2022 / Accepted: 27 June 2022

Abstract
Case-workload estimation has always been a complex process and plays a vital role in prosecutorial work. Despite the increasing development of rule-based techniques, artificial intelligence and machine learning have rarely been used to study case-workload estimation problems, leaving many cases processed without quantitative estimation. This paper aims to develop a new case-work estimation method that combines artificial intelligence methods with practical needs and apply it to the case assignment system of the prosecutor's office. We propose a feature learning model, the improved AdaBoost model, to capture the features of cases for case grouping to estimate case workload. We first learn the case textual data based on the judicial proper noun dictionary, extract the case labels from the case information with the AdaBoost learner, and group and encode each case by fuzzy matching. Then, the extracted vital information estimates case workload based on the length of case processing time and suspects number, respectively. We conducted extensive experiments to compare the proposed method with eight baseline methods, including the traditional AdaBoost classifier, to evaluate the performance of the proposed model on a real prosecution case dataset. The experimental results demonstrate the superiority of our proposed workload estimation model.

Keywords Case-workload estimation · Cases grouping and encoding (CGE) · AdaBoost

1 Introduction

Over the past few decades, the role of information technology in the judicial field has also become increasingly prominent [1, 2]. The procuratorial authorities rely on the Unified Business Application System (UBAS) in China to receive cases, and the subsequent "case assignment" process is based on "random case assignments, supplemented by assigned case assignments". Technically, it lacks the means to estimate the difficulty of the case quantitatively. Therefore, with the wide application of artificial intelligence and machine learning in various fields [3], the research to quantify the complexity of the cases can help ensure the quality of the cases in real-time and assist with processing the cases.

The issue of data analysis in judicial field is very actual and challenging [4–6]. Some critical information of the case, such as length of case processing time and suspects, will directly affect the case handling efficiency [7]. In prosecutorial cases, the length of case processing time is the period between the acceptance of a case by the prosecutor and its completion. In addition, the suspects refer to those suspected of committing a crime in the prosecutorial case. The manual annotation method can only annotate structured data, but this method is time-consuming and labor-intensive, and cannot distinguish the same type of cases significantly and effectively. There is a lot of unstructured data in the case data information, which contains a lot of valuable key information and will directly affect the case processing process [8].

In the process of case processing, the problem of case-workload estimation is directly related to the rationality of case allocation, the performance evaluation of prosecutors, and the improvement of the incentive system. Since the judicial reform, many scholars have studied the issue of case

✉ Wei Li
liwei@cse.neu.edu.cn

✉ Dazhe Zhao
zhaodz@neusoft.com

Xin Min
minxin@stumail.neu.edu.cn

[1] School of Computer Science and Engineering, Northeastern University, Shenyang 110000, China

[2] Key Laboratory of Intelligent Computing in Medical Image (MIIC), Northeastern University, Shenyang 110000, China

Published online: 21 July 2022

workload, but the current research is only at the stage of qualitative analysis. Except for listing the specific factors included in the case workload, it does not give a method that can be used for statistics and measurement of case workload [9].

This paper analyzes the historical case processing data based on the above problems. In detail, first, we segment the case data based on the judicial noun dictionary. Then, the cases are grouped and coded with improved AdaBoost and fuzzy matching. Finally, we estimate the case workload by assigning case labels and critical information (length of case processing time and suspects). By analyzing a large number of historical case data with this grouping and case-workload estimation method, it is possible to classify cases accurately and estimate the workload of each case quantitatively. This paper provides a variety of estimation output methods for different prosecutor workloads, different prosecutor offices, and different case types, which can improve the accuracy of case-workload estimation, ensure the quality of case processing, and improve case processing efficiency.

In summary, this paper has the following contributions.

- To get better case grouping encoding results, we propose the improved AdaBoost model and combine it with fuzzy matching.
- To improve the accuracy of the case workload, we propose two effective workload estimation methods for the first time: the case-workload estimation method based on the length of case processing time and suspects.
- Experimental results on actual data show that our final model is practical for case-workload estimation and intelligent case assignment.

The rest of the section is as follows. Section 2 provides a brief review of the related work. Section 3 presents the case grouping and the case-workload estimation methods, respectively. Section 4 details the experimental results and analyses. Section 5 discussion presents further results analysis and potential limitations. Finally, Sect. 6 is our conclusion.

2 Related Work

In the process of case processing, the issue of case-workload estimation is very significant. However, it is impossible to estimate case workload intuitively by dealing with case data directly. Therefore, it is necessary to extract critical information about the case.

The process of extracting critical information about a case, which is a process of classifying essential details on a case, can be seen as a multi-classification problem. Octavia-Maria et al. support the SVM algorithm for classifying legal cases [10] and propose a method combining statistics and machine learning for predicting the outcome of French Supreme Court decisions [11]. For the case text classification problem, the literature [12] compared seven standard classification methods and concluded that the basic Bayesian and KNN methods are significantly better than the other five classification methods. In this paper, combined with previous work, we choose ensemble learning for case grouping.

With the rapid development of artificial intelligence technology, machine learning, especially deep learning, have become a new trend in the development of intelligent judicial technology in the judicial field. Robert et al. use convolutional neural networks (CNNs) to extract semantic features from case text [13], thus supporting the deep mining of case data. Although CNNs can perform well in many tasks, CNN models cannot model case summary information accurately, thus failing to achieve the case grouping encoding required in this paper. The text modeling approach in the study mentioned above is based on feature labels, and labeling features require a lot of manual work and expert knowledge. Therefore, the case classification or conviction algorithm formed on this basis is not scalable. In 2016, Liu et al. [14] proposed three different information-sharing mechanisms based on the recurrent neural network (RNN) for the text multi-classification task and achieved good results in four benchmark text classification tasks. Still, the results were general for the study of this paper.

We focus on constructing a case-workload estimation method to express the complexity of a case. The case workload can be output for different prosecutors, prosecution offices, and case types.

3 Methods

This section presents technical details, including the case grouping and encoding and case-workload estimation. As shown in Fig 1, in our estimation method, we first code the case groups based on case summaries. Then, we estimate the case workload for the cases based on the encoding information and case processing information.

3.1 Notations and Problem Formulation

Table 1 depicts the notations and critical concepts used in this paper. First, we grouped and coded cases and assigned case labels. Then, workloads are estimated based on the length of case processing and case suspects number, respectively. Formally, the task of case-workload estimation is to achieve three general objectives.

- *Case grouping and coding*: It is the essential requirement of case-workload estimation which demands the case label are assigned based on the case classification.

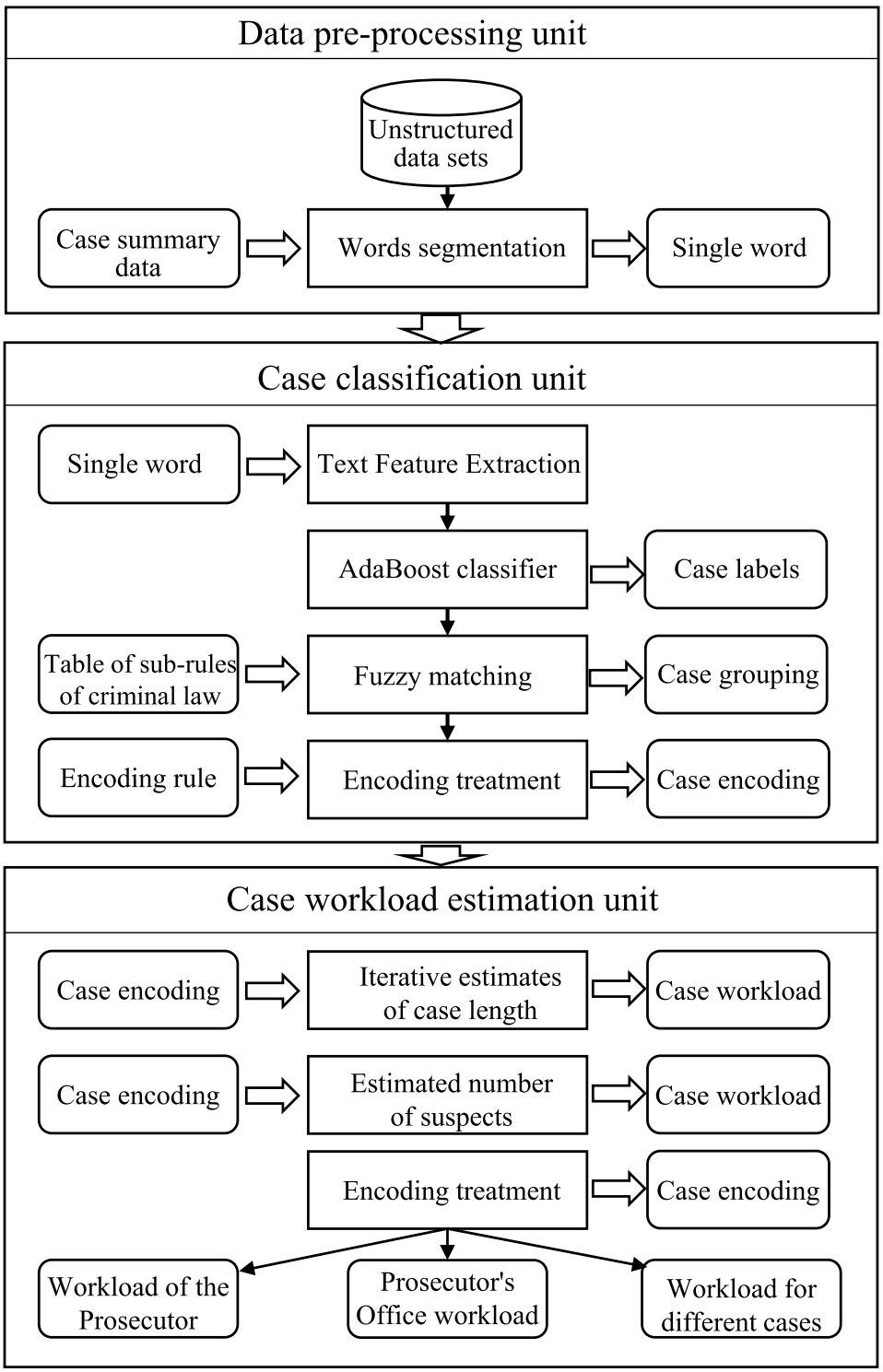

Fig. 1 Flow chart for estimating case workloads

- *Workload for length of case processing* : We estimate case workload for the length of case processing time based on case labels.
- *Workload for case suspects number*: We estimate case workload for the number of suspects based on case labels.

3.2 Case Grouping and Encoding

This section presents a case grouping encoding algorithm that assigns a case label to each case. Six months of case summary data from a particular municipal prosecutor's office is used as input. The method consists of three main

Table 1 Mathematical notations

Symbol	Description
N	The number of iterations
w_j^i	The weight set of the jth sample set at the ith iteration
W	The weight set of the sample set
X	The feature vector matrix of the dataset
Y	The set of all case labels in the dataset
h_i	The ith weak learner
a_i	The parameters of the learner h_i
β_i	The weight of the ith weak learner
P	The combination of a_i and β_i
z_i	The normalization factor
g	The workload estimation weight

Table 2 Prosecution case data

Uniform case number	*Case taker*	*Case summary*	*St*	*Et*
000-xx-x	XXX	xx ...xxx...xxxx.	xx/xx/2019	xx/xx/2019
⋮	⋮	⋮	⋮	⋮

steps. First, we pre-process case data. Then, the cases are grouped and coded with the improved AdaBoost and fuzzy matching.

3.2.1 Data Pre-processing

We collected case data from the prosecution system of a municipality that contains only criminal cases for six months, including both structured and unstructured data. Each case data consists of five main attributes in the case data. As shown in Table 2, the main qualities include case serial number, case undertaker, case summary, the start of processing time (St), the end of processing time (Et), etc., all of which are crucial for subsequent steps. A case summary data mainly contains the following information: the time of the crime, the crime, the people involved, the means of the crime, the tools of the crime, the loss of goods, the amount of loss, and so on. For example, "On January 23, 2006, Zhang San came to the sub-bureau to report that a moped parked at his home in the evening of the 22nd was stolen, valued at more than RMB 1000 yuan, in the village of xx, xx town of this district".

Specialists mainly record the case summary, which is usually maintained in textual form. These short texts contain a high number of judicial terms, resulting in problems of inaccurate case information learning.

The case summary text preprocessing in this paper is mainly for the word segmentation of the text data, which is based on the Bi-direction match method [15]. The algorithm compares the word segmentation results obtained by the forward maximal matching method with the results obtained by the reverse maximal matching method and then follows the maximal matching principle, which is to find the longest matching string as a word in the dictionary. The result with the fewest number of word segments is selected. A word assignment dictionary is a dictionary containing proper judicial nouns.

3.2.2 Case Labels Learning

Learning case labels from detailed case summary data accurately is one of the main challenging problems in case-workload computation. In contrast to traditional classification methods, we consider an ensemble learning method AdaBoost to learn case labels from case data. AdaBoost (Adaptive Boosting), was proposed by Yoav Freund and Robert Schapire in 1995 [16]. It is adaptive in the following way: the wrong samples of the previous basic classifier will be boosted, and the weighted whole samples will be used again to train the next basic classifier. At the same time, a new weak classifier is added in each iteration until some predetermined sufficiently small error rate is reached or a prespecified maximum number of iterations is reached [17–19].

Compared to the conventional AdaBoost, we presented the regularization factor when updating the weight distribution of the training set in the improved AdaBoost. The presentation of regularization factors can avoid the distribution of the extreme weights for case summary learning when the weights are updated. It is worth mentioning that we consider a prosecutor's processing of cases as a set of data, which more closely matches the following workload estimation needs and allows for easy comparison of each prosecutor's case processing capacity.

First, we ask some judicial professionals to label some cases in advance. Then, we use the already labeled data from the case data as the training set of the learner, leaving the unlabeled data as the test data set.

We input a dataset with m sample sets: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $y_i \in Y$, Y denotes the set of case labels, then we want to output the weights of sample set at the k-th iterations as follows: $W^k = (w_1^k, w_2^k, \ldots w_m^k)$. The main steps are as follows:

1. Initialize the distribution of weights for the training samples, with equal weights for each training sample.

$$W^1 = (w_1^1, w_2^1, \ldots w_m^1); w_j^1 = \frac{1}{m}; \quad j = 1, 2 \ldots m \tag{1}$$

2. Perform N rounds of iteration to produce N weak learners corresponding to $i = 1, \ldots, N$. (N predetermined):

2.1 Train weak learner by the sample set with the weight distribution W^i:

$$h_i(x;a_i) \to \Psi \tag{2}$$

where a_i denotes the parameters of the learner, h_i and Ψ denotes the learning result.

2.2 Estimate the learning error rate e_i of h_i on the training dataset.

$$e_i = P\left(X(x_j) \neq y_j\right) = \sum_{j=1}^{m} w_j^i I\left(\Psi(x_j), y_j\right) \tag{3}$$

where the function $I(\cdot)$ is expressed as $I(u,v) = \begin{cases} 1 & u \neq v \\ 0 & u = v \end{cases}$

2.3 Estimate the weight β_i of the weak learner h_i.

$$\beta_i = \frac{1}{2} \ln \frac{1 - e_i}{e_i} \tag{4}$$

2.4 Update the weight distribution of the sample dataset for the next round of $i+1$ iterations:

$$W^{i+1} = \left(w_1^{i+1}, w_2^{i+1}, \ldots w_j^{i+1}, \ldots, w_m^{i+1}\right). \tag{5}$$

$$w_j^{i+1} = \begin{cases} \frac{w_j^i}{z_i} \times e^{-\beta_i} & \Psi(x_j) = y_j \\ \frac{w_j^i}{z_i} \times e^{\beta_i} & \Psi(x_j) \neq y_j \end{cases}, \quad j = 1, \ldots, m \tag{6}$$

z_i is the normalization factor:

$$Z_i = \begin{cases} \sum_{j=1}^{m} w_j^i e^{-\beta_i} & \Psi(x_j) = y_j \\ \sum_{j=1}^{m} w_j^i e^{\beta_i} & \Psi(x_j) \neq y_j \end{cases} \tag{7}$$

3. Integrate N weak classifiers into a final strong learner.

$$F(X;P) = \sum_{i=1}^{N} \beta_i h_i(X;a_i). \tag{8}$$

where X denotes the set of feature vector x, β_i denotes the weight of the ith weak learner among the N weak learners, and P denotes the combination of β_i and a_i, $P = (a_i, \beta_i)$.

The test set feature vector X_{test} is brought into the strong learner for case label prediction. The classification process is shown in Fig 2.

3.2.3 Fuzzy Matching

Because of the need to satisfy multiple types of case-workload computation requirements, we need to assign the most relevant case category to each case label. With the development of machine learning, fuzzy matching, and fuzzy consistency studies [20–23] play an essential role in classification tasks. In this paper, we adopt FuzzyWuzzy's string fuzzy matching method for case category assignment based on the table of criminal law assignment rules [24], the main idea of which is to estimate the difference between two strings using the editorial distance (Levenshtein Distance). Therefore, here the editorial distance of two strings is denoted as follows:

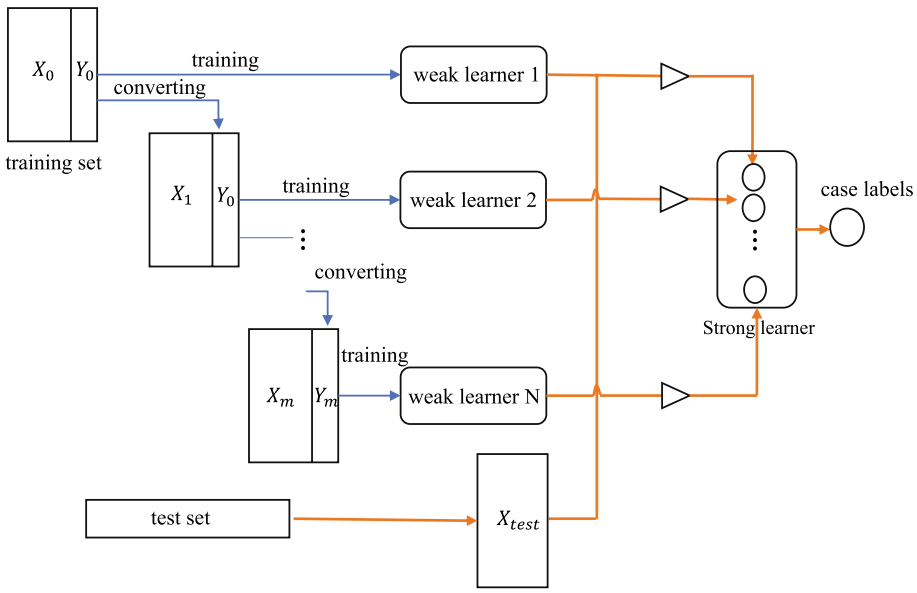

Fig. 2 Improved AdaBoost learning process

$$\mathrm{lev}_{a,b}(\mathrm{a,b}) = \begin{cases} \max(i,j) & \min(i,j) = 0 \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1 \end{cases} \end{cases} \tag{9}$$

where the Levenshtein Distance of two strings a, b is denoted as $\mathrm{lev}_{a,b}(\mathsf{a},\mathsf{b})$, where a and b denote the length of a and a, respectively. i,j denotes the i,jth character of the string, respectively. For example, a case labeled "drug offense" is fuzzily matched to a specific case type "drug offense".

3.2.4 Grouping and Encoding

To facilitate meeting the needs of subsequent case-workload estimations, we finally performed cases grouping and encoding (CGE), each CGE consisting of four digits with the following encoding rules.

- The first digit is a capital letter, and A–Y denotes the 25 first-degree counts of the case, respectively.
- The second digit is lowercase letters, with a–z denoting the secondary counts of the case, respectively.
- The third digit is an Arabic numeral indicating the order of the second-degree offense within the first-degree offense.
- The fourth digit is an Arabic number indicating whether there is a combination of crimes and the proportion of suspects.

1. *"1" indicates no combined offense and one suspect.*
2. *"3" indicates a combination of a lesser offense and a suspect.*
3. *"5" indicates a combination of an aggravating circumstance and a person suspected of committing the offense.*
4. *"4" indicates a combination of lesser offenses and not less than one suspect.*
5. *"6" indicates a combination of aggravating circumstances and not less than one person suspected of the offense. "0" indicates a case where no express indication was given.*

An example is shown in Fig 3.

For example, a CGE code Ka13, as shown in Fig. 3, is explained as follows: the case is a crime against the personal and democratic rights of citizens in the first degree (K), a crime of intentional homicide in the second degree (a), the second degree is arranged in the order of one (1) in the first degree, and the case has a combination of lesser circumstances and one suspect (3).

3.3 Case-Workload Estimation

In the previous section, each case was coded into a set of CGE codes. This section estimates the case workload based on all the CGE codes and case processing information. Among the existing case-workload research methods, they are only at the qualitative analysis stage and do not give ways that count and measure case workload. By analyzing the factors influencing a case's workload, the workload statistics and estimations for different cases based on historical case data can effectively predict the workload of unprocessed cases and provide a reference for prosecutors' workload estimation. It has been proved that the length of the case processing time and the number of suspects in the case are the main factors influencing the workload of the case. To achieve improved case-workload estimation performance, we also introduce the EM algorithm and CRF named entity technique to enhance the accuracy of case-workload estimation based on two factors. In the following, we will describe these steps in detail.

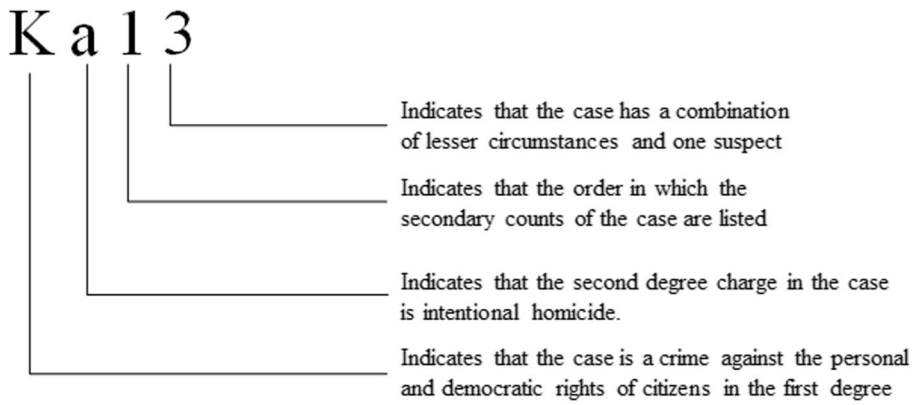

Fig. 3 Case grouping encoding design model

3.3.1 Case-Workload Estimation Based on the Length of Case Processing Time

The number of cases handled by a certain prosecutor over a period of time n, the start time of the case St_i, the end time $Et_i, i = 1, 2, \ldots, n$, the start time and end time for sorting, sorting, and stored in the array $t[j]$, $j = 1, 2, \ldots, 2n$. The time is divided into $2n - 1$ time interval, expressed as $\mathrm{d}t_j = t[j + 1] - t[j]$.

To improve the accuracy of case-workload estimation based on the length of case processing time, we propose an EM algorithm-based method to iteratively solve the case workload based on the length of case processing time. The main processes are as follows.

1. Initial workload estimation: Based on the number of crossover cases m_j for each time interval $\mathrm{d}t_j$, estimate the average case time $T_j = \frac{\mathrm{d}t_j}{m_j}$ for the interval with an initial weight of $g_j^0 = \frac{1}{m_j}$. For the i-th case, according to the start time St_i and end time Et_i, find the corresponding $t[u]$, $t[v]$, $1 \le u < v \le 2n$, and then estimate the initial workload of the case.

$$T_i^0 = T_u + T_{u+1} + \cdots + T_{v-1}. \tag{10}$$

2. Iterative computing workload: Set the number of iterations N to start the iteration for the d-th iteration.

(a) Update the weight of the case i within the case processing time interval.

$$g_{ip}^d = \frac{T_i^{d-1}}{SUM^{d-1}}, \tag{11}$$

where $SUM^d = \sum_i^{m_j} T_i^d$ represents the sum of the workload in the d-th iteration of the crossover case.

(b) Update the case workload of the case i:

$$T_i^d = \left(\mathrm{d}t_u \times g_{i1}^d\right) + \left(\mathrm{d}t_{u+1} \times g_{i2}^d\right) + \cdots + \left(\mathrm{d}t_{v-1} \times g_{iM}^d\right). \tag{12}$$

workload factor:

$$\alpha_i^N = \frac{T_i^N}{\sum_i^n T_i^N}. \tag{13}$$

To fully understand estimating the case workload for this algorithm, the case-specific workload is estimated for cases where three of the time dimensions intersect, as shown in Fig 4.

I. Estimation of initial weights: $g_1^0 = g_2^0 = g_3^0 = 1$
II. Estimation of initial workload:

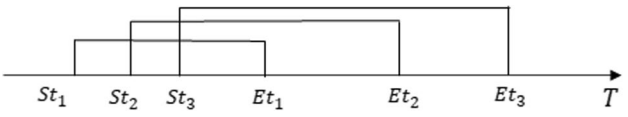

Fig. 4 Case timeline

i. Initial workload for case 1: $T_1^0 = T_{St_1 \to St_2} + \frac{1}{2}T_{St_2 \to St_3} + \frac{1}{3}T_{St_3 \to Et_1}$
ii. Initial workload for case 2: $T_2^0 = \frac{1}{2}T_{St_2 \to St_3} + \frac{1}{3}T_{St_3 \to Et_1} + \frac{1}{2}T_{Et_1 \to Et_2}$
iii. Initial workload for case 3: $T_3^0 = \frac{1}{3}T_{S_1 \to Et_1} + \frac{1}{2}T_{E_{t_1} \to Et_2} + T_{Et_2 \to Et_3}$

where the $T_{St_1 - St_2}$ represents the period between the moment of commencement of Case 1 and the moment of commencement of Case 2 on the timeline. The $T_{St_2 \to St_3}$ represents the period between the moment of commencement of Case 2 and the moment of commencement of Case 3 on the timeline. The $T_{St_3 \to Et_1}$ represents the period between the moment of commencement of Case 3 and the moment of termination of Case 1 on the timeline. The $T_{Et_1 \to Et_2}$ represents the period between the moment of termination of Case 1 and the moment of termination of Case 2 on the timeline. The $T_{Et_2 \to Et_3}$ represents the period between the moment of termination of Case 2 and the moment of termination of Case 3 on the timeline.

We set the number of iterations $N = 10$, substitute the initial workload, and start the iterative estimation. Then, the weight and workload for the z-th iteration are as follows:

1. Weight of case 1: $g_1^z = \frac{T_1^z}{T_1^z + T_2^z + T_3^z}$
2. Weight of case 2: $g_1^z = \frac{T_1^z}{T_1^z + T_2^z + T_3^z}$
3. Weight of case 3: $g_3^z = \frac{T_3^z}{T_1^z + T_2^z + T_3^z}$

(a) Workload of case 1: $T_1^z = T_{S_1 \to St_2} + T_{St_2 \to St_3} * \frac{g_1^{z-1}}{g_1^{z-1} + g_2^{z-1}} + T_{St_3 \to E_1} * \frac{g_1^{z-1}}{g_1^{z-1} + g_2^{z-1} + g_3^{z-1}}$

(b) Workload of case 2: $T_2^z = T_{St_2 \to St_3} * \frac{g_2^{z-1}}{g_1^{z-1} + g_2^{z-1}} + T_{St_3 \to Et_1} * \frac{g_2^{z-1}}{g_1^{z-1} + g_2^{z-1} + g_3^{z-1}} + T_{E_1 \to Et_2} * \frac{g_2^{z-1}}{g_2^{z-1} + g_3^{z-1}}$

(c) Workload of case 3: $T_3^z = T_{St_1 \to Et_1} * \frac{g_3^{z-1}}{g_1^{z-1} + g_2^{z-1} + g_3^{z-1}} + T_{Et_1 \to Et_2} * \frac{g_3^{z-1}}{g_2^{z-1} + g_3^{z-1}} + T_{Et_2 \to Et_3}$

Based on the above method, it is possible to estimate the workload per case based on the length of case processing time $f(x) = T_x^n$.

3.3.2 Case-Workload Estimation Based on Suspects

After getting the case workload based on the length of time, we estimated the case workload of criminal suspect. In this paper, CRF named entity recognition technology is used to extract criminal suspects. Conditional Random Fields (CRF) is a conditional probability distribution model for another set of output sequences given a set of input sequences, which is widely used in natural language entity naming and recognition[25].

The whole flow chart is as follows (Fig 5):

With the above method, we first identify and extract the suspects of each case in the dataset, then eliminate the repeated suspects in the cases, and finally count the number of suspects, such that $h(x) = N_x$, where x is the case ordinal number.

3.3.3 Case-Workload Estimation

Case-workload estimation mainly includes two parts: case-workload estimation based on the length of case processing time and criminal suspect. After estimating the case workload based on the case processing time and the criminal suspect, the two kinds of workloads are weighted and summed: case workload $F(x)$:

$$F(x) = u_1 f(x) + u_2 h(x). \tag{14}$$

The u_1 and u_2, respectively, represent the weights of the two workloads, according to the experimental analysis of previous data, which are set as $u_1 = 0.7, u_2 = 0.3$.

4 Analysis of Experimental Results

In this part, experiments verify the effectiveness of the proposed method in case-workload estimation. We first describe the data set and then validate our approach. Finally, we will analyze the validity of our case-workload estimation method.

4.1 Dataset

We adopted a real dataset extracted from a city prosecutor's office whose cases cover many common cases. The dataset is a collection of data on criminal cases in the municipality for 2018–2019. A criminal case is a case in which a suspect is accused of violating a social relationship protected by the Criminal Law. The state investigates, tries, and imposes criminal sanctions to hold the suspect or defendant criminally responsible for the crime. Detailed statistics are summarized in Table 3. The statistics on the length of case processing time (LOC) and the number of suspects (NOS) allow us to infer that the workload of various criminal cases varies. After data preparation, the data are approximately 1000 criminal case data.

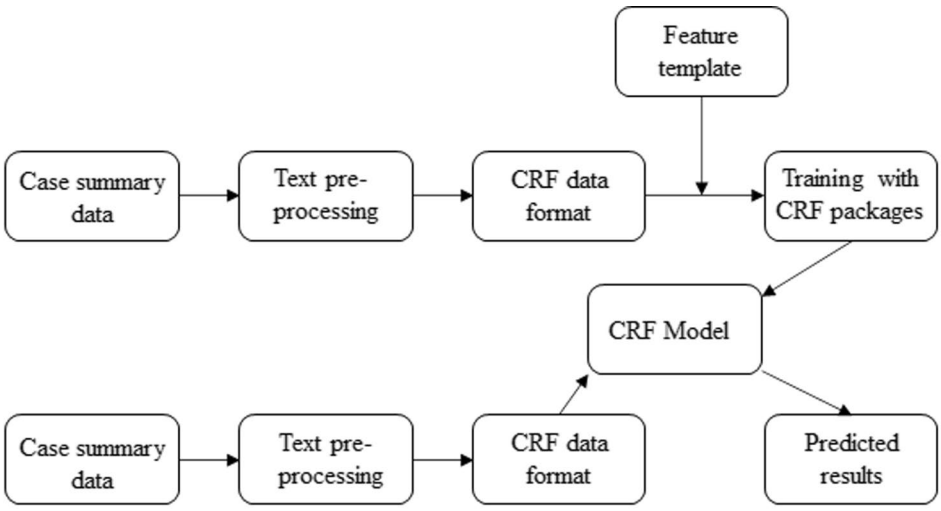

Fig. 5 Extraction of criminal suspects based on CRF technology

Table 3 Statistics of our dataset

Number of cases	Avg LOC	Min LOC	Max LOC	Avg NOS	Min NOS	Max NOS
1032	7	1	36	4	1	10

4.2 Comparison of Case Grouping Results

To evaluate the performance of improved AdaBoost in our data set, we implemented seven classic classification models and analyzed and compared the accuracy of grouping.

Compare the accuracy, respectively:

$$\mathrm{Acc} = \frac{\#h}{|X_{\text{test}}|}, \tag{15}$$

where $\#h$ denotes the number of test sets that are correctly classified and $|X_{\text{test}}|$ indicates the number of all test sets.

Precision:

$$P = \frac{1}{K}\sum_{i=1}^{K} P_i = \frac{1}{K}\sum_{i=1}^{K} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i}, \tag{16}$$

Recall:

$$R = \frac{1}{K}\sum_{i=1}^{K} R_i = \frac{1}{K}\sum_{i=1}^{K} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i}, \tag{17}$$

where K is the number of categories, TP, FP, TN, and FN indicate the number of true positive, false positive, true negative, and false negative, respectively, and the subscript i indicates which category it belongs to.

Although the evaluation metrics of accuracy and recall are good, they are usually in conflict with each other, and improving one usually comes at the expense of the other. Therefore, we combine precision and recall to introduce the F1 value. The F1 value is estimated as follows:

$$F_1 = \frac{2PR}{P+R}. \tag{18}$$

The experimental setup and evaluation methods are fair for all comparisons, and we focus on the relevant improvements taken in this paper. The experimental results are shown in Table 4. The bolded values in Table 4 are the results of our method, and the same is true for Table 5.

As can be seen from Table 4, under a consistent experimental setup condition, the method proposed in this paper was obviously superior to the traditional classification methods. Since traditional methods mostly use a single learner, AdaBoost was a kind of ensemble learning, which can accumulate the advantages of multiple learners. Due to the limited amount of prosecutorial case data, it was difficult to train an accurate model with deep learning such as CNN. Moreover, compared to the best baseline method AdaBoost, our method showed better performance in terms of evaluation metrics. It proves that our improvement of AdaBoost with regularization factors is effective.

Furthermore, we also compared the case grouping and encoding results. As shown in Table 5, we first asked the professionals to manually group and code the 1032 cases. We grouped and coded the 1032 cases based on the classification results of the baseline methods and proposed method. It was observed that the proposed model exhibited better performance and improved the accuracy of case grouping and encoding.

4.3 Case-Workload Analysis and Comparison

The case workload was analyzed for multiple factors (different prosecutors, different prosecution offices, and different case types), and the results are shown in Figs. 6 and 7.

5 Discussion

It is feasible to use the case data to group and code, then the case labels are assigned to each case, and finally, the case workload is estimated from the length of the case processing and the number of suspects. Compared with the traditional method, it has the following characteristics.

First, the cases are grouped and coded by an improved AdaBoost classifier, therefore, whose effect of grouping is better than a single classifier.

Second, since the length of case processing time and suspects are the most important factors affecting the complexity of cases, it is very reasonable to utilize them to estimate the

Table 4 Comparison of grouping effects

Classification method	Accuracy	Precision	Recall	F1
Naive Bayes	0.814	0.792	0.785	0.794
KNN	0.823	0.801	0.796	0.785
Decision Tree	0.793	0.794	0.786	0.788
SVM	0.806	0.792	0.786	0.794
RNN [14]	0.697	0.712	0.694	0.696
Logistic regression	0.722	0.797	0.787	0.785
MLP	0.769	0.758	0.762	0.755
AdaBoost [16]	0.809	0.793	0.828	0.817
The proposed	**0.857**	**0.844**	**0.862**	**0.856**

Table 5 Comparison of grouping and encoding results

Classification method	Correct number	Error number	Precision
Naive Bayes	824	208	79.8%
KNN	856	176	82.9%
Decision Tree	803	220	78.5%
SVM	816	216	79.1%
RNN [14]	720	312	69.8%
Logistic regression	748	284	72.5%
MLP	793	239	76.8%
AdaBoost [16]	831	201	80.5%
The proposed	**880**	**152**	**85.2%**

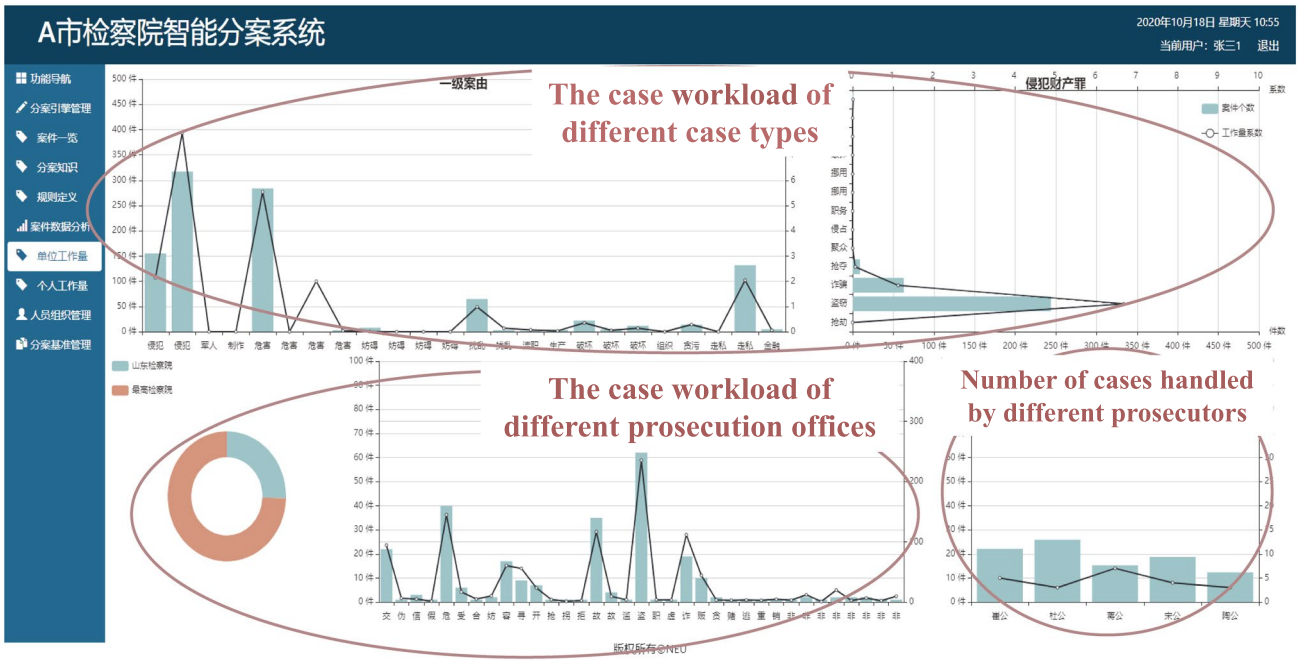

Fig. 6 Case-workload comparison of different procuratorates and different types of cases

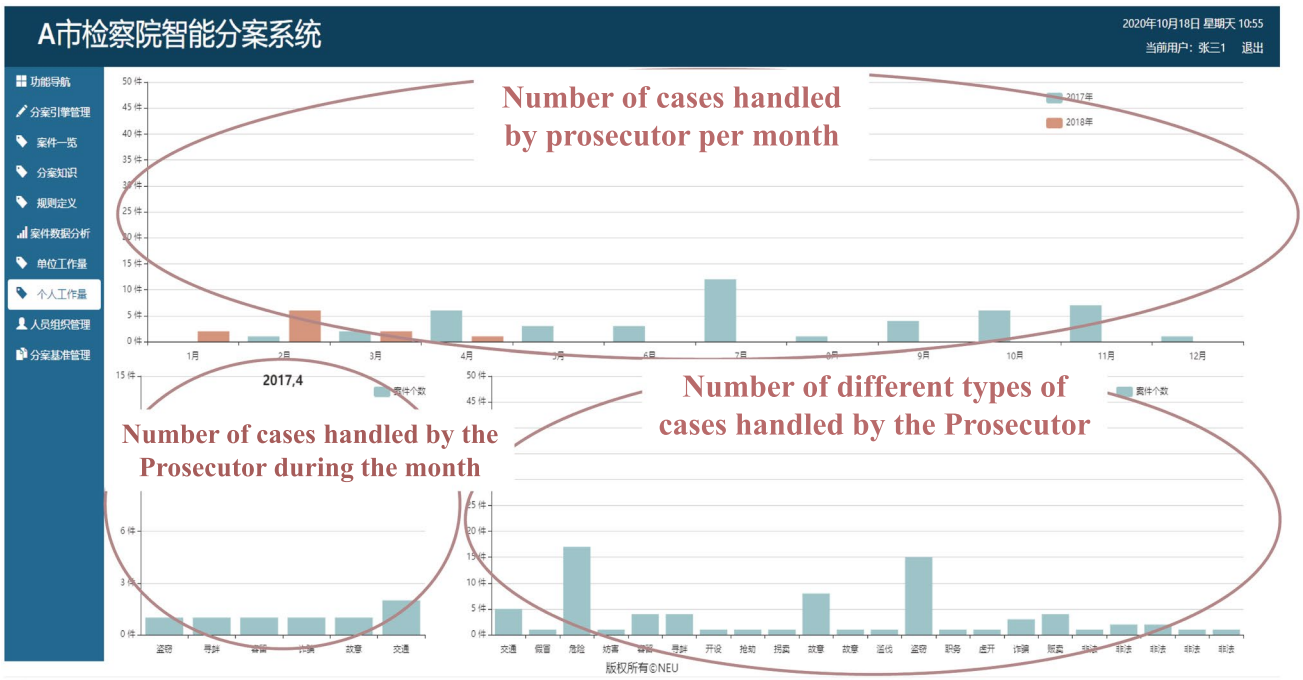

Fig. 7 Estimation and statistics of prosecutors' personal workload

case workload. It also provides a new idea for quantitative evaluation of complex cases in the judicial field.

Finally, the case assignment system based on case workload is more efficient than the traditional random assignment strategy. The satisfaction of case assignments is also greatly improved.

However, during the development of this method, a few potential limitations need to be considered.

Table 6 Abbreviations list

Abbreviations	Description
UBAS	The unified business application system
CGE	The cases grouping and encoding
CRF	The conditional random fields
LOC	The length of case processing time
NOS	The number of suspects
CNNs	The convolutional neural networks
SVM	The support vector machine
KNN	The k-nearest neighbor

1. For datasets with too small whit a sample size, it may have the possibility of overfitting.
2. We have mainly exploited the length of case processing time and the number of suspects to estimate the case-load. However, prosecution case workload is influenced by several case features in practical application, such as the number of case files involved and case ratio. Our future research work will attempt to leverage multiple case features to improve the accuracy of case-workload estimation.

6 Conclusion

This paper developed a case-workload estimation technique for judicial research, which provides an effective tool for prosecutors' offices to assign cases rationally. Specifically, we first adopted an improved AdaBoost for grouping and coding prosecutorial cases to determine the category labels of cases. Then, we estimated the case workload with the length of the case processing and the number of suspects in the case. We conducted extensive experiments to evaluate the performance of the proposed model on an actual legal case dataset. The experimental results demonstrated the superiority of the proposed model in this paper.

The judicial scenario is one of the essential components of practical application scenarios, and our research provides new light on intelligent justice. Further, our research will focus on multi-features case-workload estimation, brilliant assignment of prosecutorial cases, etc.

The list of abbreviations is shown in Table 6.

Acknowledgements We acknowledge the project The National Key R &D Program of China (No. 2021YFC2701003), and Fundamental Research Funds for the Central Universities (N2016006) for supporting our work.

Author Contributions XM wrote the main manuscript, WX and JY prepared figures and tables. WL primarily provided data and methodology. DZ supervised the whole process. All the authors reviewed the manuscript.

Funding The National Key R &D Program of China (No.2021YFC2701003), Fundamental Research Funds for the Central Universities (N2016006).

Data Availability The data used in this paper are from the real case data of the prosecutor's office, and so it cannot be made freely available. Requests for access to these data should be made to the corresponding author.

Declarations

Conflict of Interest The authors declared that they have no conflicts of interest to this work.

References

1. Lodder, A.R., Gordon, T.F.: The pleadings game: An artificial intelligence model of procedural justice. Artif. Intell. Law **8**(2/3), 255–264 (2000)
2. He, T.-K., Lian, H., Qin, Z.-M., Chen, Z.-Y., Luo, B.: Ptm: a topic model for the inferring of the penalty. J. Comput. Sci. Technol. **33**(4), 756–767 (2018)
3. Aggarwal, K., Mijwil, M.M., Al-Mistarehi, A.-H., Alomari, S., Gök, M., Alaabdin, A.M.Z., Abdulrhman, S.H.: Has the future started? The current growth of artificial intelligence, machine learning, and deep learning. Iraqi J. Comput. Sci. Math. **3**(1), 115–123 (2022)
4. Chen, D.L.: Judicial analytics and the great transformation of American law. Artif. Intell. Law **27**(1), 15–42 (2019)
5. Yu, S., Chen, X.: How to justify a backing's eligibility for a warrant: the justification of a legal interpretation in a hard case. Artif. Intell. Law 1–30 (2022)
6. Mandal, A., Ghosh, K., Ghosh, S., Mandal, S.: Unsupervised approaches for measuring textual similarity between legal court case reports. Artif. Intell. Law **29**(3), 417–451 (2021)
7. Van Rhee, C.H.: (ed.) Judicial case management and efficiency in civil litigation. Antwerp/Oxford: Intersentia, vol 70 (2008)
8. Dai, F., Song, Y., Si, W., Yang, G., Hu, J., Wang, X.: Improved CBSO: a distributed fuzzy-based adaptive synthetic oversampling algorithm for imbalanced judicial data. Inf. Sci. **569**, 70–89 (2021)
9. Šavelka, J.: Coherence as constraint satisfaction: Judicial reasoning support mechanism. Coherence: Insights Philos. Jurisprud. Artif. Intell. Springer, Dordrecht, pp. 203–216 (2013)
10. Şulea, O., Zampieri, M., Malmasi, S., Vela, M., Dinu, L.P., Genabith, J.van.: Exploring the Use of Text Classification in the Legal Domain.in Proc. ASAIL, London, UK, pp. 1–5 (2017)

11. Şulea, O., Zampieri, M., Vela, M., Genabith, J.van.: Predicting the lawarea and decisions of french supreme court cases. In: Proc. RANLP, Varna, Bulgaria, pp. 716–722 (2017)
12. Wei, F., Qin, H., Ye, S., Zhao, H.: Empirical study of deep learning for text classification in legal document review. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 3317–3320 (2018)
13. Keeling, R., Chhatwal, R., Huber-Fliflet, N., Zhang, J., Wei, F., Zhao, H., Shi, Y., Qin, H.: Empirical comparisons of CNN with other learning algorithms for text classification in legal document review. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 2038–2042 (2019)
14. Liu, P., Qiu, X., Huang, X.: Recurrent neuralnetwork for text classification with multi-task learning. In: IJCAI, pp. 2873–2879 (2016)
15. Fanjin, M.: Sense disambiguation of Chinese segmentation based on bi-direction matching method and HMM. New Technol. Lib. Inf. Serv. **24**, 37–41 (2008)
16. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: ICML, vol. 96, pp. 148–156 (1996)
17. Onan, A., Korukoğlu, S., Bulut, H.: A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. Expert Syst. Appl. **62**, 1–16 (2016)
18. Majidpour, H., Gharehchopogh, F.S.: An improved flower pollination algorithm with adaboost algorithm for feature selection in text documents classification. J. Adv. Comp. Res. **9**(1), 29–40 (2018)
19. Onan, A.: Hybrid supervised clustering based ensemble scheme for text classification. Kybernetes **46**(2), 330–348 (2017)
20. Arqub, O.A., Al-Smadi, M.: Fuzzy conformable fractional differential equations: novel extended approach and new numerical solutions. Soft. Comput. **24**(16), 12501–12522 (2020)
21. Abu Arqub, O., Al-Smadi, M., Momani, S., Hayat, T.: Numerical solutions of fuzzy differential equations using reproducing kernel Hilbert space method. Soft. Comput. **20**(8), 3283–3302 (2016)
22. Arqub, O.A., Al-Smadi, M., Momani, S., Hayat, T.: Application of reproducing kernel algorithm for solving second-order, two-point fuzzy boundary value problems. Soft. Comput. **21**(23), 7191–7206 (2017)
23. Abu Arqub, O.: Adaptation of reproducing kernel algorithm for solving fuzzy Fredholm-Volterra integro differential equations. Neural Comput. Appl. **28**(7), 1591–1610 (2017)
24. Pei, W.: Criminal reconciliation in China: consequentialism in history, legislation, and practice. China-EU Law J. **3**(3), 191–221 (2014)
25. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282–289 (2001)