ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 134

2nd International Conference on Control, Automation, and Artificial Intelligence (CAAI 2017)

A Universal Method for Intelligent Judgement

Mingzhe Li*, Hongli Zhang, Lin Ye and Chuanwang Ma

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

*Corresponding author

***Abstract*—In this paper, a universal method is proposed for intelligent judgement, which relies on feature vectors representing each case to enable intelligent judgement via machine learning algorithms. The process to extract feature vectors consists of three main steps: modeling the case, building feature words lists, and extracting the vectors. After feature vectors are built, kNN and SVM algorithms are used to train the classification model, and the performance is evaluated through the experiments.**

Keywords-intelligent judgementt; model; feature vectors; machine learning; svm

I. INTRODUCTION

Artificial intelligence (AI) is widely developed in many fields, such as fingerprint identification, sentiment analysis, intelligent control, etc. Besides these mentioned, AI has played an important role in more traditional fields over the last decade [1].

Nevertheless, there are few mature methods using AI in judicial process. The judge and jury hear all the evidence given by both sides, investigate the truth, and make the court decision based on the laws [2]. In this process, AI can provide a suggestion for reference, by an intelligent model trained by big data.

In fact, intelligent judgement for various cases can be regarded as a type of text classification problem essentially. The judge makes the conclusion by features of the case. Therefore, natural language processing (NLP) and machine learning are necessary. Many machine learning algorithms proved well in many situations, such as k-Nearest Neighbor (KNN) and Support Vector Machine (SVM). Inspired by the observation, these two methods can be used to train the classification model by large quantities of previous judicative documents, and compare the performance between them. In order to get better models, manual intervention plays an important role in the process of generating training data.

In this paper, the method to extract feature vectors for each case is described first. The method includes modeling the case, building feature words list, and extracting the vectors. The conclusion of judicative case is treated as labels. The training data and test data consists of feature vectors with labels. Then SVM and KNN models are trained and tested. Finally, we compare two models through the experiments, and make the conclusion about this paper.

II. FEATURE VECTORS FOR JUDICATIVE CASE

In this method, the data in the form of N-dimensions vectors are required, for training classification model. To extract the feature vectors of each judicative document, a universal method including three steps is proposed: modeling the case, building the feature words list, and extracting the vectors. The generated vectors are used to train the text classification model. The specific descriptions are as follows.

A. Modeling Judicative Case

Judicative documents record the process of trial and the result of judgement. So the key objects of intelligent judgement are these documents crawled from public websites, where the key features can be extracted. According to our analysis, the documents can be divided into four parts, including basic information, facts, affirmation, and conclusion [3]. Each document can be described by a vector, so the vector set V consists of n vectors (n is the number of documents in database).

$$V = \{V_i | i = 1, 2, 3 \dots n\} \tag{1}$$

The vector is defined as follows:

$$V_i = (I, F, A, C) \tag{2}$$

In (2), I represents basic information, consisting of case information I_{case}, and involved party information I_{party}:

$$I = (I_{case}, I_{party}) \tag{3}$$

In (2), F represents facts, consisting of the cause of action F_{reason}, and the appeal of party expressed by F_{appeal}.

$$F = (F_{reason}, F_{appeal}) \tag{4}$$

In (2), A means affirmation, consisting of $A_{identification}$ and $A_{influence}$. $A_{identification}$ are the facts identified by court. $A_{influence}$ are the elements which influence judgement.

$$A = (A_{identification}, A_{influence}) \tag{5}$$

In (2), C is conclusion, consisting of c_{base} and C_{add}. c_{base} is the outline of conclusions, and C_{add} is a supplement of trial conclusion.

$$C = (c_{base}, C_{add}), c_{base} \in \{0, 1\} \tag{6}$$

The model is universal for various types of cases. This process is named ‘first-layer features extraction’.

In these equations, a factor is represented by lowercase if it can be described by a real number directly.

The judgement conclusion is influenced by facts and affirmation. So when it goes to a certain type of case, the 'second-layer features extraction' is necessary. The extraction depends on legal contexts, so it varies from case to case.

In this paper, divorce case is studied as an example.

By studying legal context as well as consulting professional lawyers, the second-layer features extraction method is as follows.

For F_{reason}, infidelity, violence, bad habits, separation are considered in legal contexts.

$$F_{reason} = (i, v, b, s)\ d, v, b \geq 0, s \in \{0,1,2\} \quad (7)$$

Where i is infidelity, v is domestic violence, b is bad habits, and s is separation. First three factors increase with severity.

For F_{appeal}, the appeals are almost about property and custody besides getting a divorce.

$$F_{appeal} = (p, c)\ p, c \in \{0,1\} \quad (8)\backslash$$

Where p is property, and c is custody.

For A, some factors absent in law contexts still influence the judgement in reality. Such as minor children, the attitude of defendant and whether the appeal is for the first time.

$$A_{influence} = (k, a, f) \quad (9)$$

$$k, f \in \{0,1\}, a \in \{-1,0,1\}$$

Where k is whether there is a minor child, and f is whether it is the first appeal.

In summary, a divorce case can be modeled by a n-dimensional vector, according to the 'two-layer' extraction. The vectors are built to train machine learning model, and c_{base} extracted from judicial document is treated as label.

The vector consists of i, v, b, s, p, c, k, a, f. The i-th case document in database is expressed as follows:

$$V_i = (i, v, b, s, p, c, k, a, f) \quad (10)$$

The attribute and explanation of each dimensionality are shown in Table I.

TABLE I. DIMENSIONALITY DECL EXPLANATION

Items	Meaning	Range	Explanation
i	Infidelity	$[0, +\infty)$	increase with severity
v	Violence	$[0, +\infty)$	increase with severity
b	Bad habits	$[0, +\infty)$	increase with severity
s	Separation	{0,1,2}	0 means no/not referred, 1 means separation time is less than 2 years, 2 means separation time is more than 2 years.
p	Property	{0,1}	0 means no property dispute, 1 means existing.
c	Custody	{0,1}	0 means no custody dispute, 1 means existing.
k	Minor child	{0,1}	0 means no minor child, 1 means existing.
a	Attitude	{-1,0,1}	-1 means against, 0 means indifferent, 1 means agreed
f	Fisrt time	{0,1}	0 means it is the first time of appeal

B. Building Feature Words Lists

To extract vectors, feature words lists are fundamental. With the help of word2vec, feature words lists are built first.

Word2vec is a tool that can map words into k-dimensionality real value vectors to simplify the calculation process, developed by a deep learning team of google [4]. By means of training, text contents operation is translated into vectors operation. The similarity in vector space can be used to represent the similarity in text semantics. Therefore, the term vectors generated from word2vec are useful in many NLP fields, such as clustering [5].

The linguistic model of word2vec is a kind of hierarchical Log-bilinear model, including CBOW and Skip-Gram. The former model is based on context to predict the intermediate word:

$$P(w_t | w_{t-k}, w_{t-(k-1)}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}) \quad (11)$$

Skip-Gram is the opposite of CBOW, predicting context based on a word [6].

In this paper, CBOW is used. After term vectors are built, the feature words lists are generated by clustering these vectors. We choose a type, then the word list of this type is gained.

The words lists contain two matrices, named first-level words list and second-level words list. First-level words list is a N*k matrix, M_1. N is the number of dimensions of feature vectors. k is the number of key words for each feature. Each row in M_1 consists of the key words to represent this feature as concise and comprehensive as possible. For example, $M_1[0]$ is a list including all key words about infidelity (i in (10)), such as 'adultery', 'mistress', etc.

The second-level words list is a N*j matrix M_2.N is the number of dimensions of feature vectors as well. j is the length of key words dictionary for each feature. Unlike the former, M_2 [0] is a dictionary including the key words and corresponding weight about infidelity (i in (10)). Words in the dictionary can be classified to content words and degree words. The content words describe the feature, the weights are less

than 1; degree words are adjective or frequency, the weights are greater than or equal to 1. Take the factor i as an example, the words such as 'ambiguity' is a content word, the weight is 0.2, and the weight of 'mistress' is 0.4, showing the latter is more serious; the word such as 'occasionally', 'long-term' are degree words, of which the weights are 1 and 1.2.

More detailed differences and usages will be expressed in next section.

C. *Extracting the Vectors*

The feature vector has two main types of dimensions: the first one is described by different values, which show the severity degree, such as i, v, b, s; the second one is described by 0 or 1, expressing whether the factor exists, such as p, c, k, a, f. Given the situation, a mixed algorithm is designed to extract feature vectors.

1) *For the former type of dimensions*:

The dimensionalities are quantitative. So the algorithm is based on two matrices ($M_1 M_2$) generated from last section. For a judicial document, we handle it by Chinese word segmentation first, and then match each line in the document with matrix M_1. Once the line has a key word in $M_1[m]$, the line should compare with the m-th row in matrix M_2 next. Each row in M_2, is a dictionary for the certain feature, consisting of various words about the feature with each weight. The max weight of common content words in this and $M_2[m]$ will multiply by max weight of common degree words, and the result is added to previous value of certain feature finally. Each line in document has the chance to match the whole M_1, and the value is stored for each dimensionality in the feature vector of each document.

The pseudocode is shown in Table Ⅱ.

There are some additional explanations:

- Once the line in document has key words in $M_1[m]$, the m-th dimensionality in feature vector has an initial weight value (greater than 1).
- Each line in document has the chance to influence feature vector. So the weight of certain dimensionality is cumulative for the same judicial document.
- To avoid special circumstances, the influence-times threshold of each dimensionality for the case is set. Once a dimensionality has been added for more than threshold , the value of this dimensionality will not be changed, regardless of what follows in the document.

2) *For the latter type of dimensions:*

The dimensionalities are qualitative. So the extracting method is to determine whether the factor exists or not. Given the demand, first-level words list is remained, of which the usage is similar. Moreover, the regular expressions are prepared for solving this problem. The regulations are extracted from documents. For example, the statement contains 'second', 'once', 'ever', 'appeal', if it is about whether this appeal is for the first time. As for the attitude of defendant, relevant statement contains 'agree', or 'disagree', 'divorce' and so on. Under this approach, the left dimensionalities are extracted.

TABLE II. THE PSEUDOCODE FOR EXTRACTING FEATURE VECTORS

Algorithm: EXTRACT FEATURE VECTORS

input: judicial documents sets F, M_1, M_2
output: case feature vectors sets R
1. for each judicial document F_i in sets F:
2. for each line L in document F_i:
3. if L has key word in $M_1[m]$, m ← 0 to N:
4. if L has content word/degree word in $M_2[m]$:
5. if $M_2[m][content\_ward]$ > max- content:
6. Max- content ← $M_2[m][content\_ward]$
7. if $M_2[m][degree\_ward]$ > max_ degree:
8. max_ degree ← $M_2[m][degree\_ward]$
9. $R_i[m] \leftarrow R_i[m]$ + max_ content * max_ degree
10. return R

3) *Normalization:*

Because of various types, the ranges are different from dimensionality to dimensionality. When the unprocessed vectors are obtained for all case documents in database, normalization is required.

There are two methods to normalize the data. The one is min-max normalization (12), the other is Z-score normalization (13).

$$x^* = \frac{x - min}{max - min} \tag{12}$$

$$x^* = \frac{x - \mu}{\sigma} \tag{13}$$

Where μ is the mean of sample data, and σ is the standard deviation of sample data. After normalization, the mean of results is 0, as well as the variance is 1.

Compared with min-max normalization, Z-score normalization has better performance in similarity measurement. So Z-score normalization is selected.

So far, the available vectors are prepared. Each case is described by a n-dimensions vector. As mentioned, c_{base} is treated as label, to classify the vectors. In this case, c_{base} is whether the divorce is granted.

III. CLASSIFICATION MODEL

In this section, two algorithms are introduced. By the algorithm, the machine learning model is trained to classify the new case. The classification results are the results of intelligent judgement. The intelligent judgement process relies on manual intervention, and the clusters of results are few. So KNN and SVM algorithms are needed.

A. *K-Nearest Neighbor Algorithm*

KNN is a basic algorithm to classify the data. The core concept of KNN is that a sample data belongs to the same classification if most of k-nearest neighbors belong to a classification. The result depends heavily on k-nearest neighbors [7].

B. Support Vector Machine

Given training data set D, finds a hyperplane to divide different data in sample space [8].

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}, y_i \in \{-1, +1\} \quad (14)$$

In sample space, partition hyperplane can be described by a linear equation as follows:

$$\omega^T x + b = 0 \quad (15)$$

Where $\omega = (\omega_1; \omega_2; \ldots; \omega_d)$ is a normal vector, determining the direction of hyperplane; b is the displacement term, determining the distance between hyperplane and origin.

The target is to find a partition hyperplane which has maximum margin [9].

However, not all sample space has a partition hyperplane to deal two-class classification problem. The sample should be mapped to a higher-dimensional characteristic space, to make sample data linearly separable. The kernel function plays an important role in mapping process. Different kernel functions are suited for different situations. Generally, radical basis function behave better on many issues.

IV. EXPERIMENTS

The data is crawled from Chinese Judicative Documents Website (wenshu.court.gov.cn). The type of cases is divorce. The number of judicative documents is 5922 totally. The feature vector is extracted from each document, and after normalization, the training data and test data is gained. The labels of data are 'divorce granted' and 'divorce rejected'. According to the statistics, the number of label '+1' which means the divorce is granted is 2256, and the number of label '-1' is 3666, which means the divorce is not granted.

The training data and test data is divided by 4:1. And the training data and test data hold the same ratio between label '+1' and label '-1'.

The experiments make use of libsvm [9] and scikit-learn library, based on python in Windows 10.

The result of experiment for KNN is shown in Figure Ⅰ.

Figure Ⅰ shows the accuracy reaches max value: 90.33%, when k is 1. Given no noise data in data set, k=1 means the new case belongs to the same classification with the most similar case. The accuracy is less as increasing k. When k is 2, the accuracy is 88.83%. And when k is 44, the accuracy is only 80.92%.

For SVM, the model is trained by the same training set. As shown in Figure Ⅱ, and the training process adopts cross validation, the accuracy can reach 98.1364% when C is 64 as well as gamma is 0.125.

After training, the model is tested by test data. The accuracy is 97.0833%, which means 1165/1200 cases are judged right in this intelligent method.

Through the experiments, SVM shows a better performance than KNN on the divorce.

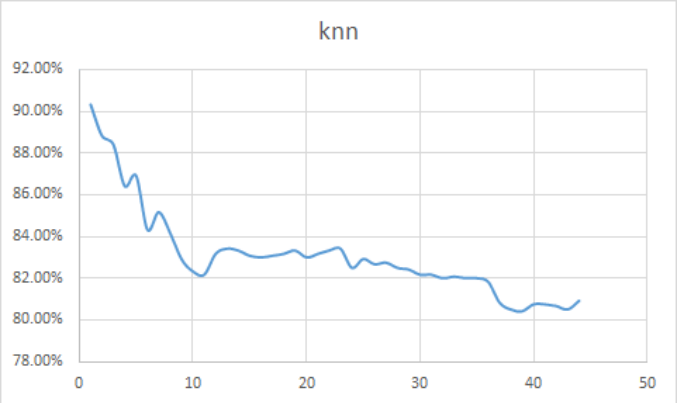

FIGURE I. ACCURACY BY K FOR KNN

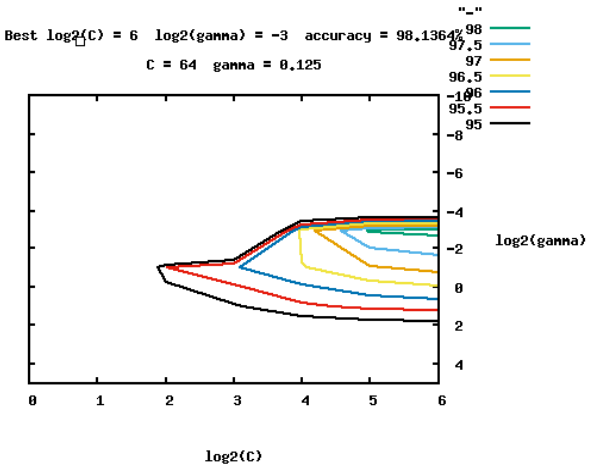

FIGURE II. ACCURACY BY C AND GAMMA FOR SVM

V. CONCLUSION

In this paper, a universal method for intelligent judgement is proposed. In this method, a two-layers model is built for extracting features from various cases. The first-layer model is based on document structure, and the second-layer model relies on certain law provision. Then building feature words lists, and extract feature vectors by the model and lists. The results need normalization.

The experiments for KNN and SVM use the results as trained data and test data. The experimental results show that SVM has a better performance on this issue, in term of accuracy.

REFERENCES

[1] mt.sohu.com/2016/1209/n475347884.shtml

[2] Wan-Chen Lin, Tsung-Ting Kuo, Tung-Jia Chang, Chueh-An Yen, Chao-Ju Chen et al, "Exploiting Machine Learning Models for Chinese Legal Documents Labeling, Case Classification, and Sentencing Prediction," Computational Linguistics and Chinese Language Processing Vol. 17, No.4, December 2012 pp. 49-68

[3] Liu, C-L., Chang, C.-T.. & Ho, J.J. “Case Instance Generation and Generation and Refinement for Case-Based Criminal Summary Judgments in Chinese,” Journal of Information Science and Engineering, 20, 783-800 2004.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. “Efficient Estimation of Word Representations in Vector Space,” in Proceedings of Workshop at ICLR, 2013

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. “Distributed Representations of Words and Phrases and their Compositionality,” in Proceedings of NIPS, 2013.

[6] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. “Linguistic Regularities in Continuous Space Word Representations,” in Proceedings of NAACL HLT, 2013.

[7] Trevor Hastie and Rolbert Tibshirani. “Discriminant Adaptive Nearest Neighbor Classification,” in IEEE TRANSACTIONS ON PAITERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 18, NO. 6, JUNE, 1996.

[8] Chang, C.-C. and C.-J. Lin.(2011).”LIBSVM:A library for support vector machines.”ACM Transactions on Intelligent Systems and Technology, 2(3):27, 2011 .

[9] https://github.com/cjlin1/libsvm