Journal of High School Science

Efficient court justice using auto-judgement with Artificial Intelligence

Liu P[1], Liu J[2], Liu J[3], Ding E.X[4]

Submitted: October 1, 2023, Revised: version 1, October 9, 2023, version 2, February 10, 2024
Accepted: February 11, 2024

Abstract

It is vital to legal practitioners and citizens in the community that legal judgements handed down by the courts be impartial, just and efficiently executed. The absence of these attributes could exacerbate and create, existing and new social problems respectively. In this work, we used Artificial Intelligence to model an assistant system to automatically predict the judgement of court cases based on the description of the facts. We applied natural language processing (NLP) and deep neural networks (DNNs) to predict court judgments. We used and analyzed the public dataset from the US Supreme Court for its essential characteristics. We then performed vectorization (TF-IDF) to convert the judgement facts into input data and they were used for our AI model built by 1D convolutional neural networks. We performed data cleaning and resampling to reduce data noise and imbalance. In addition, dropout and network simplification were adopted to avoid overfitting. Our AI model demonstrated a > 95% accuracy when predicting the result of a subset of cases from the US Supreme Court.

Keywords

Common law system, Facts analysis, Natural language processing, Deep Neural networks, Data cleaning, Data imbalance, Overfitting, Court judgements, Artificial Intelligence, Supreme court

[4]Corresponding author: Emily X. Ding, Vineyards AI Lab, Auckland, New Zealand, 0630. emily.ding@vineyardsailab.com

[1]Phoebe (Yun) Liu, liucloud0515@gmail.com, [2]Josephine (Hsin) Liu, josephyuliu@gmail.com, [3]Joseph (Yu) Liu, hsinliu0515@gmail.com, Rangitoto College, 564 East Coast Road, Mairangi Bay, Auckland, New Zealand, 0630 and Vineyards AI Lab, Auckland, New Zealand, 0630

1 Introduction

Legal experts who make judgments in court cases need specialized knowledge and experience in the field of Law and in related fields. Arriving at a decision involves several steps, such as searching relevant law articles/history cases, defining the range of the charge, deciding the penalty term, etc (1-2). The entire process is time-consuming and requires a solid domain-specific background, burdening the limited amounts of legal practitioners. Recent reports and articles state that legal institutes in most countries suffer from significant delays due to the large numbers of cases and the deficiency in human resources (3). This causes courts to take months, if not years, to decide cases. For example, only ~100-150 of 7,000 Supreme Court cases are appropriately reviewed (4). 1.43% of court cases pending in the Supreme Court are heard yearly. It is understandable that reviewing and analyzing complicated plea documents and fact descriptions takes a lot of workforce and time. The restrictions and limitations to a human, including energy, time, memory capability, and most importantly, the ability to be fair in the face of temptations, suggest the desperate need for another option, system, or a helping hand. A recent report showed that 4.1% of the defendants sentenced to death in the United States in 2014 were actually innocent, raising questions about the judges' motives and whether the justice carried out was genuinely fair (5). In addition, many people who need legal help spend a disproportionate amount of time to receive it. The process of legal assistance and/or court decisions has become insufficient, lengthy, inefficient, unfair and costly. These reasons were our motivation to create a helping hand to law institutions with a view to providing more efficient, righteous, timely, affordable and correct legal judgements and/or decisions.

To create such a helping hand, Artificial Intelligence (AI) technology is the best candidate. AI is anticipated to lend significant support to Law institutions and the Supreme Court to analyze and store data, provide reasoning and logical results, and have the ability to keep and learn from resources and data within the field. We decided to choose the challenging benchmark dataset in the paper by Alsayed et al. (6), which is well-annotated using data from the US Supreme Court. The objective was to construct a solution that could predict the judgment results using the facts from the plaintiff and the defendant. Our research was generally organized as follows: First, some background about litigation and the law system is described. Next, we analyzed the dataset's characteristics and developed the prediction scheme based on deep learning. Specifically, we employed data cleaning, resampling, dropout, and network simplification technologies to optimize our solution and achieve greater accuracy. Finally, the potential applications and limitations of the model are discussed.

2 Background

2.1 Basic knowledge of legal judgement

Around the world, there are four basic types of legal systems: civil law, common law, customary law and religious law, or combinations of these. The distribution of the various types of legal systems are shown in the map in Figure 1 (7).

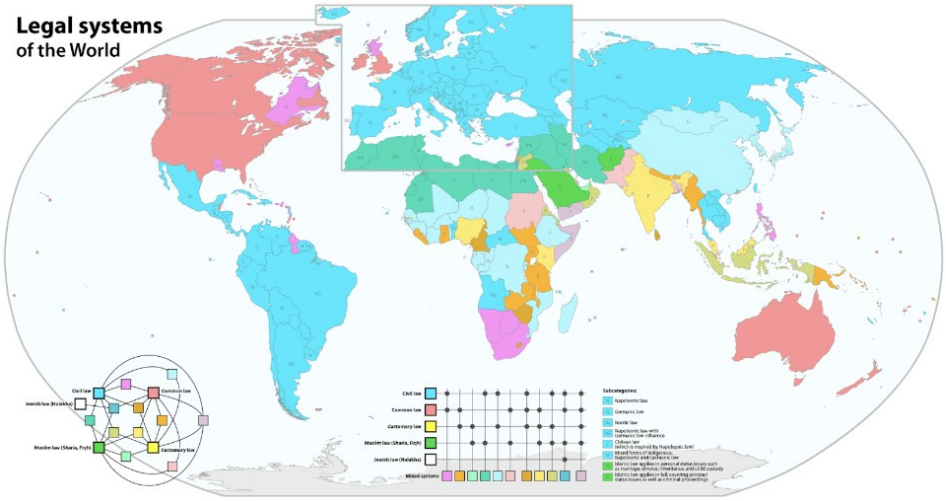


Figure 1. Legal Systems of the World (Wikipedia)

Figure 1 shows that civil-law and common-law systems are the two significant legal systems used in the world. The former is used in countries like Germany, France, and China, whereas the latter is used in Anglo-Saxon countries such as the United Kingdom, the United States, Australia, Canada, and New Zealand. The two legal systems have the same goals of bringing harmony and order to society, but they differ in their chosen methods (2). As shown in Figure 2, there are three main parts to the procedure for court judgments of both systems.

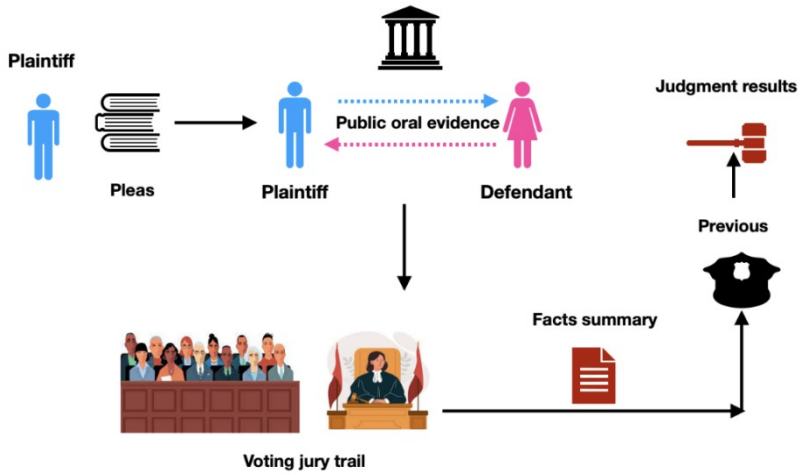


Figure 2. The procedure of court judgement

In the pre-trial stage, an investigation is carried out in order to judge if there is enough evidence for the prosecutor to press charges against the defendant. In common law, if there is sufficient evidence obtained at the pre-trial claim stage (investigation), the prosecutor is given the choice of whether or not they wish to prosecute the case albeit there being the essential evidence needed. Whereas, in civil law, if the evidence is secured at the end of the investigation, there is an obligation for the prosecutor to go through with the case. If the prosecutor decides to press charges, they will enter the

second stage of court trials, wherein lawyers will debate the cases of the prosecutor and defendant in a formal court in front of the judge. Furthermore, common law requires witnesses to give their evidence before the court, while civil law requires the witness to provide written proof in private before the trial, after which the judge questions them before the court in a neutral standing. In addition, the common law's verdict is based on the Jury's voting, but the Jury are not required to explain or justify their verdict. In comparison, the civil law judge must provide evidence to support their decision with factual reasoning based on the facts given, the statutory laws, and their own common sense. After the trial, if the written law regarding the case is clear and there is no doubt or ambiguity regarding the verdict, common law judges follow their provisions. In the possibility wherein doubt and ambiguity do surface, judges will search for similar past cases and be guided accordingly and apply to the current case appropriately, taking into consideration the rules of law, evidence, and the facts determined by the Jury.

2.2 Natural language processing and judgement in court

Natural language processing aims to build a system to process and understand human language. It is a field at the intersection of computer science, artificial intelligence, and linguistics (8). Law and the general court system is a closed and self-sufficient system in logic. That means the law provides some standards or rules for the facts displayed in the court, which guide judgment decisions. There hence exists a fortuitous similarity between these sets of rules in Artificial Intelligence (AI) or machine learning and those that guide the court. Therefore, the scientific community has tried to develop auto-judgement using AI and NLP to increase the efficiency of legal institutes, legal situations, court justice, and more legal issues in and out of the court. By following and adhering to the sense of legal reasoning used in the courts, we believe that the machine can draw a fair conclusion. Figure 3 displays the relationship between AI and court decisions.

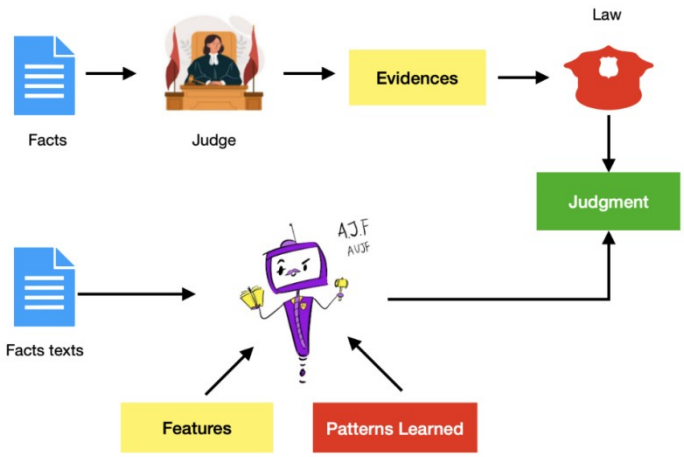


Figure 3. Relationships between court judgement and AI assistant judgement

Our objective was to develop a model that could accurately predict the results of a case using the facts provided by both the petitioner and respondent, without bias. The judges and

juries can use the results as a reference for this "justice" adhering to the logic. It could be used for minor cases or to give a general idea of the Supreme Court's decision. The tasks could be described as legal judgment prediction (LJP), which predicts the decision of legal cases based on the fact descriptions (9). There are several public studies about automated LJP in different languages such as English, Chinese, French, Korean and Indian, among others (10-12). Niklaus et al. (13), have evaluated the performance of the LJP on a multilingual 21-year dataset.

As high school students who have studied and lived in New Zealand, we selected an English and common-law system dataset for our research. With the motivation of learning AI technologies and solving practical problems, we started our auto-legal judgement prediction project, and evaluated the performances in various settings.

3 Our solution for the auto-judgement system

In this section, our solutions and research trajectory are described. First, we investigated the publicly available datasets and selected the dataset investigated by Alsayed (6), which is well annotated and pertains to legal documents from the Supreme Court of the US. It could be regarded as a similar scenario to New Zealand, which is an English country with a common-law system. We then performed a critical analysis for data evaluation from three aspects. According to the evaluation results, we designed an auto-legal judgement system employing data cleaning, feature resampling, and other related methods to avoid overfitting using 1D convolutional neural networks. Our auto-legal judgement system framework is shown in Figure 4.

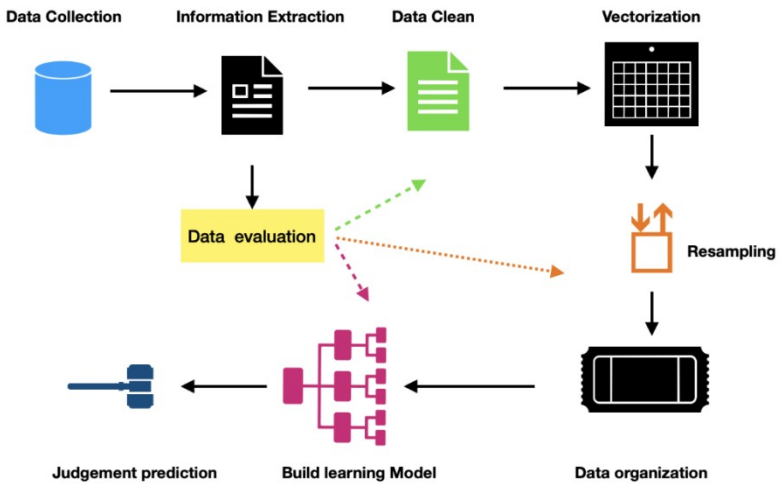


Figure 4. Our auto-legal judgement system

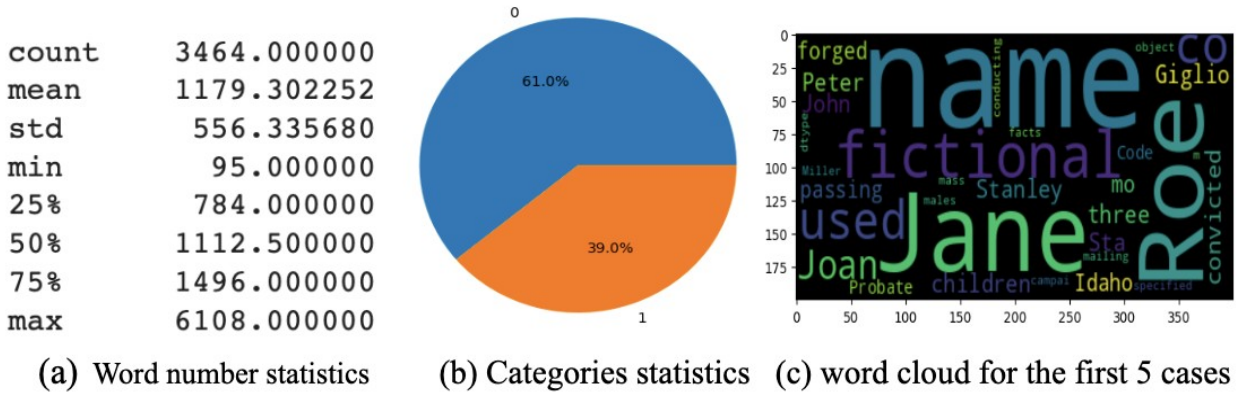
3.1 Data evaluation

Figure 5 shows some samples in the dataset. It includes information such as ID, name, href, first /second party, facts, winning party, and winner_index. In our solution, facts were the primary analysis targets, and winner_index were the labels.

level_0	index	ID	name	href	first_party	second_party	winning_party	winner_index	Facts
0	0	50606	Roe v. Wade	https://api.oyez.org/cases/1971/70-18	Jane Roe	Henry Wade	Jane Roe	0	In 1970, Jane Roe (a fictional name used in court documents to protect the plaintiff's identity) filed a lawsuit against Henry Wade, the district attorney of Dallas County, Texas, where she resided, challenging a Texas law making abortion illegal except by a doctor's orders to save a woman's life. In her lawsuit, Roe alleged that the state laws were unconstitutionally vague and abridged her right of personal privacy, protected by the First, Fourth, Fifth, Ninth, and Fourteenth Amendments.
1	1	50613	Stanley v. Illinois	https://api.oyez.org/cases/1971/70-5014	Peter Stanley, Sr.	Illinois	Stanley	0	Joan Stanley had three children with Peter Stanley. The Stanleys never married, but lived together off and on for 18 years. When Joan died, the State of Illinois took the children. Under Illinois law, unwed fathers were presumed unfit parents regardless of their actual fitness and their children became wards of the state. Peter appealed the decision, arguing that the Illinois law violated the Equal Protection Clause of the Fourteenth Amendment because unwed mothers were not deprived of their children without a showing that they were actually unfit parents. The Illinois Supreme Court rejected Stanley's Equal Protection claim, holding that his actual fitness as a parent was irrelevant because he and the children's mother were unmarried.
2	2	50623	Giglio v. United States	https://api.oyez.org/cases/1971/70-29	John Giglio	United States	Giglio	0	John Giglio was convicted of passing forged money orders. While his appeal to the you.S. Court of Appeals for the Second Circuit was pending, Giglio's counsel discovered new evidence. The evidence indicated that the prosecution failed to disclose that it promised a key witness immunity from prosecution in exchange for testimony against Giglio. The district court denied Giglio's motion for a new trial, finding that the error did not affect the verdict. The Court of Appeals affirmed.

Figure 5. Some examples in the dataset

Figure 5 shows that when winner_index = 0, the first_party is the winner, and the second_party is the winner when winner_index=1. In addition, the text format "Facts" is the most critical factor in making the final decision. Figure 6 shows some characteristics of this dataset, such as word number statistics, categories statistics, and word cloud.



(a) Word number statistics (b) Categories statistics (c) word cloud for the first 5 cases

Figure 6. Word number characteristics. (a) Category statistics, (b) Category statistics, (c), Word cloud for the first 5 cases

3.1.1 Volume of data

The volume of data in the dataset is a total of 3464 cases, which, when compared with other similar experiments, is an acceptable number. We assigned 20% of the data for testing or validation and assigned the remaining 2771 cases for training. The mean number of words was ~ 1179. This number represents a medium data size in NLP models.

3.1.2 Imbalanced categories

The proportion of label 0 and label 1 was 61% and 39%, respectively. The number of label 0 was about 1.56 times as label 1, and hence, the data was imbalanced. Data imbalance results when the data obtained has

a high concentration of data from one class (the majority class) and a low concentration from another (the minority class). This problem creates a false accuracy. Since the model cannot accurately identify and classify the minority group, it stays biased toward the majority group (14).

3.1.3 Noise and redundancy

Meaningless data, often called data noise, are the unimportant and unused words that often appear in documents. Figure 6 (c) expresses a word cloud for the first 5 examples in the dataset, in which the larger the font the word is in, the higher its frequency of occurrence, and consequently, the lower its probability of contribution to the analysis and result. From the word cloud, it can be seen that "names" are the highest frequency words and will not contribute too much. In addition, there are some redundancy symbols, errors, and incorrect information. They could result in the model making predictions of greater uncertainty (15).

3.2 Information Extraction and Data Cleaning

In the common-law system, the facts are the primary evidence for the judgement. They are the primary source of informational input for our model. After extracting the facts, data cleaning is a necessary step. Data cleaning aims to ensure that the data is correct, consistent, and usable (16). It can be implemented by identifying errors or corruptions, correcting or deleting, etc. The data cleaning module started with normalisation, which converted all text data into lowercase letters. The HTML tags were removed, since that is a demonstrated source of the noise produced when collecting text data using web or screen scraping. The special characters, symbols, numbers, and multiple spaces were also expunged. In addition, lemmatization was performed to obtain the word's root form. The output of lemmatization is the root word called lemma. The lemma of the words “Am”, “Are”, “Is” is “Be”, and the output is “go” for the words “went”, “go”, “going”, “goes”, “gone”.

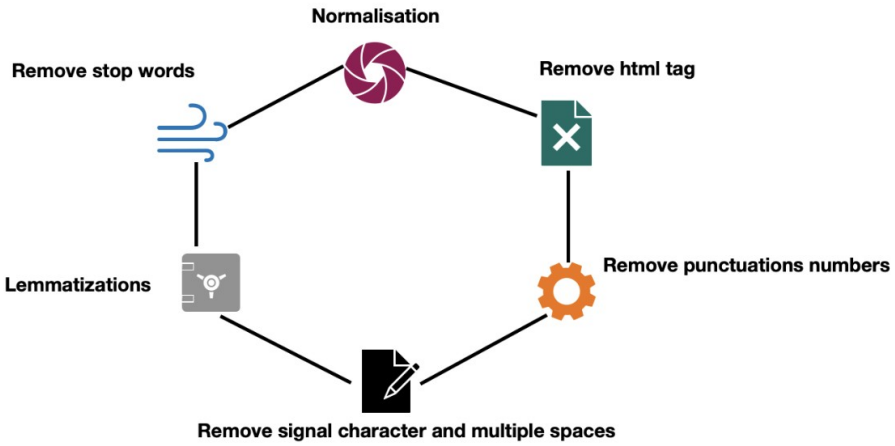


Figure 7. Data cleaning pipeline in our method

The data cleaning module ended with the removal of stop words, which are words that do not hold any specific meaning in the text, such as "is", "and", "or", etc. The stop words list was combined in different languages, and we used the English language from the Natural Language Tool Kit (NLTK) library. The word cloud in Figure 6 shows that the

first party and second party's names were also seen as "stop words" and were hence removed.

3.3 Vectorization and resampling

A "clean" text-based dataset was obtained using the methods described above. However, most models cannot directly process and understand strings or plain text. They require numerical vectors as inputs. We hence converted the text data into numerical data, known as vectorization. One of the simplest and most popular vectorization methods is Term frequency times inverse document frequency (TF-IDF). This is a statistical method that aims to quantify the importance of a given word relative to other words in the document and the corpus (8). Mathematically, there are two quantities, TF and IDF, which can be combined to arrive at the TF-IDF score.

$$TF-IDF\left(X_{-!i}(d)\right)=TF_d\left(X_{-!i}(d)\right)\times IDF\left(X_{-!i}(d)\right) \quad (1)$$

TF (term frequency) is used to measure how often a term or word occurs in the target document or text and is computed as $TF_d\left(X_{-!i}(d)\right)=\frac{N_d}{N_{-!i}}$. Where N_d is the number of occurrences of the term d in the *ith* fact. $N_{-!i}$ is the total number of terms in the *ith* fact.

And IDF (inverse document frequency) measures the importance of the term across a corpus and can be calculated as $IDF\left(X_{fact-i}(d)\right)=\log\frac{l+1}{l_d+1}+1$, where l is the total number of documents in the corpus and l_d is the number of documents with the term d.

One of the ways to overcome the problem of imbalanced data and obtain better accuracy, is to upsample the minority data after vectorization. We choose to upsample the minority class instead of downsampling the majority class data to not reduce the dataset further. The objectcive of upsampling the data is to create similar/identified data artificially to boost the amount of data of the minority class. This was performed using the Synthetic Minority Oversampling Technique method (SMOTE). SMOTE was proposed by Chawla et al. in 2002 (17). The minority class was oversampled by generating synthetic examples based on the feature space. Specifically, SMOTE synthesises new samples between existing samples using linear interpolation. These new samples are generated by selecting one or more of their closest neighbours to be classified as one of the minority classes. The principle of SMOTE is presented in Figure 8, the new samples can be represented using equation (2).

$X_s=X_0+w\left(X_1-X_0\right)$ (2) Where w is a uniform random variable in the range $[0,1]$. After the oversampling processes, the data was reconstructed to be more balanced.

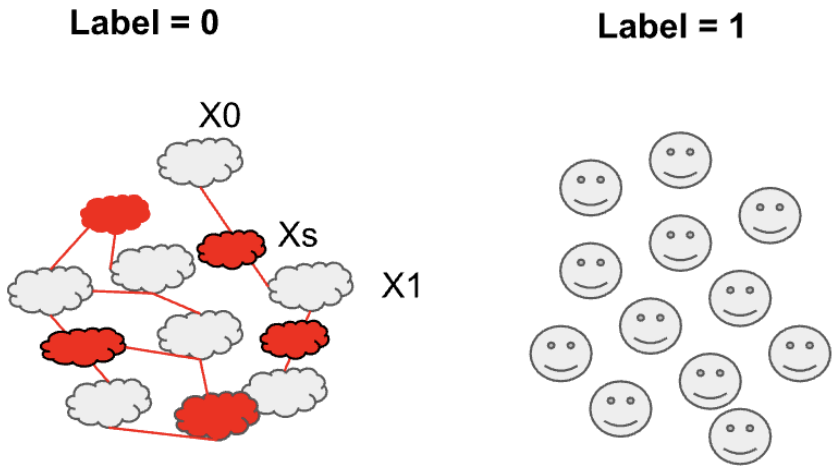


Figure 8. The principle of SMOTE

3.4 Deep Neural Networks

Deep learning is an outstanding achievement in the field of AI. Artificial neural networks simulate the working nervous system. After learning the basic MLP model, 1D convolutional neural networks (18) were employed. The basic architecture of the network is shown in Figure 9.

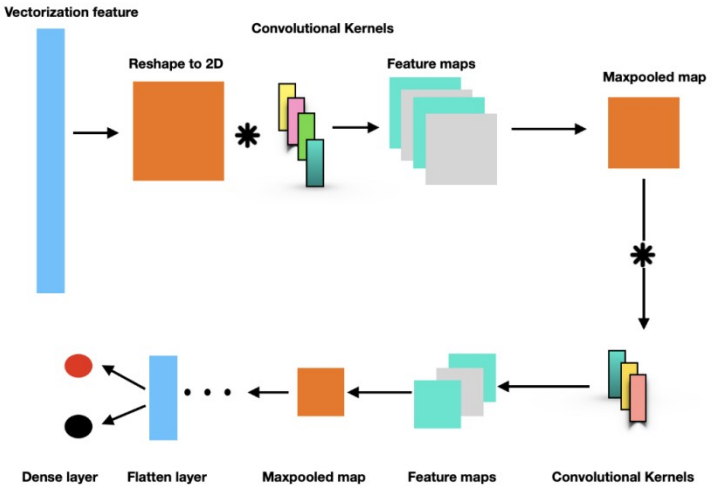


Figure 9. 1D Convolutional neural network

After vectorization, each case was displayed by a 1D vectorization feature. First, we reshaped it to 2D data and make it fit the neural networks, which is the input for the deep neural networks.

There are two basic blocks in the 1D convolutional neural networks. The first one is convolution Block, which consists of convolution and pooling layers. This layer forms the essential component of feature extraction. The other one is the fully connected block, consisting of a fully connected simple neural network architecture. This layer performs the

classification task based on the convolutional block's output.

To understand the convolution block, we present the padding, convolution, and Maxpooling in Figure 10. Convolution is a special operation applied to a particular matrix (reshaped 2D vectorization feature). The process involves multiplying the values of a cell corresponding to a specific row and column of the 2D vectorization matrix with the value of the corresponding cell in the convolutional kernel. This operation is repeated on all convolutional kernels and added together to form an output.

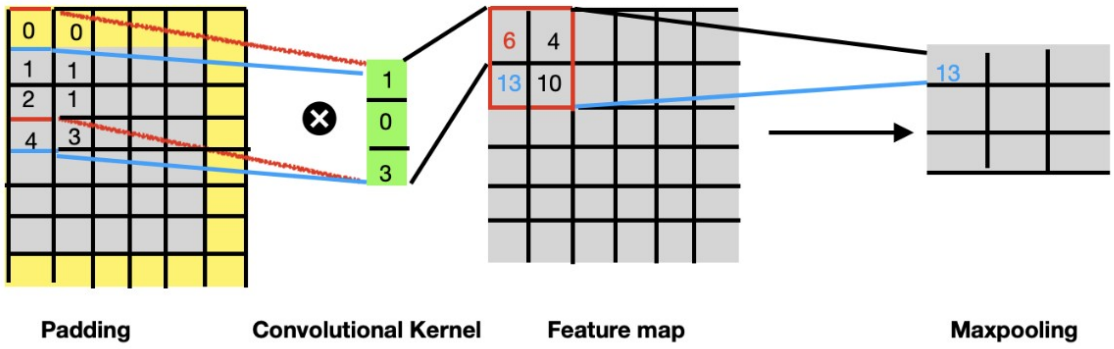


Figure 10. Convolutional Blocks details

In addition, there are some essential operations in Convolutional blocks: padding, striding, and maxpooling. Padding represents the yellow colored row and column in Figure 10. Padding adds another row and column and fills them with zeros. When we apply the convolution kernel on the input 2D matrix, the output map will be smaller, and some information may be lost at the borders of the input. Therefore, padding is used to preserve the information at the borders. Second, striding is a parameter that decides the kernel movement on the map (example, 2 or 3 rows or columns) each time. Finally, a pooling layer is employed to reduce the size of the output. Maxpooling refers to selecting the maximum value in a window of size X, and this window is slid over the input with a stride of length S after each pooling operation.

We represent the convolution blocks output mathematically, and the first is the convolution between kernel and feature map, take the *lth* layer as the example

$$y^l=\sum_{k=1}^{N_k}[b_k^l+conv1D\left(w_k^l,S^{l-1}\right)] \quad (3)$$

Where y^l is convolution result, b_k^l is the bias from k^{th} kernel in layer l, S^{l-1} is output from layer $l-1$, w_k^l is the k^{th} convolutional kernel in layer l.

Then the convolution result will pass the activation function $f(.)$ as follows,

$x^l=f\left(y^l\right)$ (4), We selected the rectified linear activation function (ReLU) in this work, which is a non-linear function, where

$f(x)=max(0,x)$. That means that if it is positive, the function will output the input directly, otherwise, it will output a zero.

The last step**,** max-pooling is employed for the down-sampling operation and can be represented as $S^l=x^l\downarrow s$ (5), Where S^l is the output of the layer l and $\downarrow s$ represents the down-sampling with the window size s, such as $(2\times2)\vee(3\times3)$.

Finally, the output of the last convolution blocks are flattened which converts the 2D feature maps to a single dimensional vector. The flattened vector is the input of dense layers for classification tasks, represented as z, where the SoftMax function is employed. The SoftMax function returns the probability of each class and its equation is shown as follows:

$$softmax(z_i)=\frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (6)$$

Here, z_i represents the valuate from the *ith* neurons of the output layer.

3.5 Overfitting

In addition, a common problem in training models called overfitting is considered, which occurs when the model arrives at accurate predictions for training data, but not for unseen data. In other words, the model tries to achieve high accuracy and ends up focusing too much on the noise and details within the training dataset (Supreme Court in the US), it leads to hyper-focusing on these details and neglecting the general pattern that is needed to be discovered by the model for it to grow (19). Thus when the model is used for other datasets (e.g., Supreme Court in NZ), the results are inaccurate.

To improve the model's generalization, some methods can be introduced to avoid overfitting, such as early stopping, dropout, network reduction, regularization and data augmentation, among others (20, 21). Considering the practical applications in the future, dropout, and network simplification were explored in our model.

To illustrate the concept of overfitting, consider an example when a bird needs to be recognized as being a bird. Usually, the model learns the essential feature of a bird. However, due to overfitting, it focuses on a specific characteristic, such as the blue feather (Figure 11). As a result, when a bird with a red feather is input into the model, it struggles to identify it as being a bird.

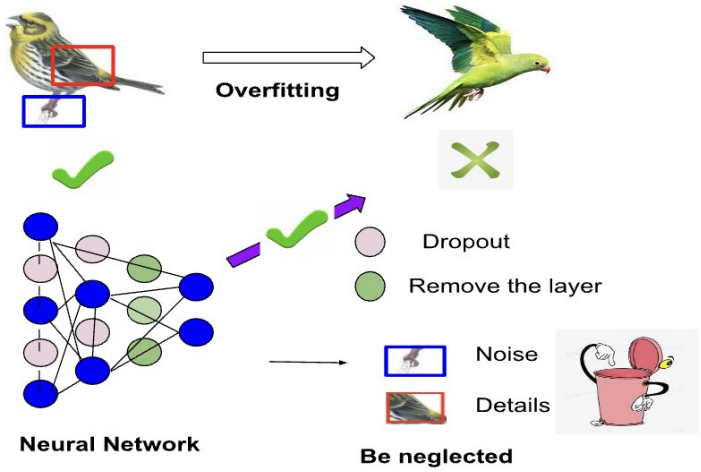


Figure 11. Overfitting and our solutions

To address overfitting, two methods were introduced. The first is dropout, where neurons in the model are removed randomly, meaning their connections are severed. This forces the model to not rely on a single neuron but instead to diversify its approach. Consequently, the model can overcome the hyper-focus on noise and detail. The second solution is network simplification, which decreases the network complexity by removing layers or reducing the number of neurons. By eliminating some layers or reducing the number of neurons, the network has fewer parameters to learn. In this situation, the network will be forced to generalize because it does not have access to detailed attributes of the image.

4 Experimental results

4.1 Experimental Parameters Setting

To prove the effectiveness of our solution, we performed experiments and analyses alongside the presentation of the results (code of experiments can be found on the Github repository: https://github.com/Vineyards-AI-Lab/auto_judgement_nlp). We focused on the evaluations of the performance of data cleaning, resampling, and neural network settings. First, a baseline of 1D convolutional neural networks were set, whose key parameters and architectures are listed in Table 1.

Table 1. Neural network settings in the baseline model

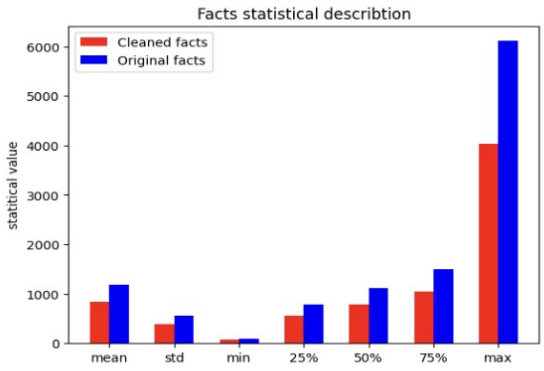
Convolutional layers	Convolutional kernel number	Kernel size	Activation functions(1D Conv)	Strides
3	1st: 64 2nd:128 3rd: 256	1st:1×5 2nd: 1×3 3rd:1×5	ReLU	1
Loss function	optimizer	Padding	Activation function (Dense)	-
Sparse categorical cross	Adam	same	SoftMax	-

4.2 Data cleaning and prediction accuracy

Data cleaning is a tool used to correct inaccurate data and records. Figure 12 shows the changes before and after data cleaning in our model.

count	3464.000000
mean	832.166859
std	384.345237
min	74.000000
25%	562.750000
50%	793.000000
75%	1040.000000
max	4037.000000

(a) Words statistics after cleaning



(b) Changes before and after data cleaning

Figure 12. Data cleaning results

A comparison of Figure 12(a) with Figure 6(a) shows that the mean of word numbers decreased from 1179 to 832. Nearly 2000 characters are removed from this particular case. According to the statistics, about 20%-30% of characters are deleted after data cleaning. Therefore, reducing the data noise would mean the system could compute faster with less workload. The prediction accuracy is displayed in Figure 13.

The results indicate a similarity between the raw and cleaned data, with an accuracy of 93.36% for the raw data and an accuracy of 93.22% for the cleaned data. In addition, errors occur more when inputting the samples labeled as a 1st party win. The error rate of nearly 10.3% is likely attributable to data imbalance.

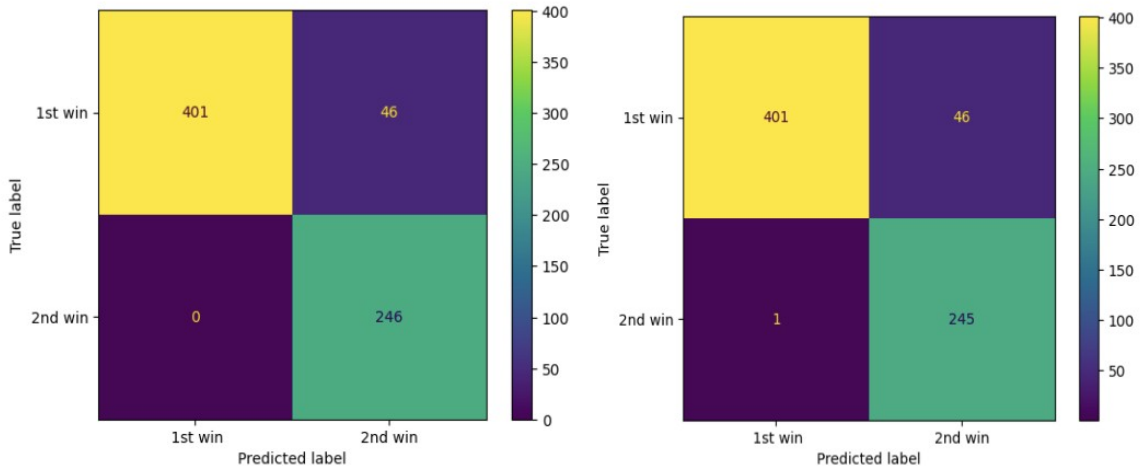


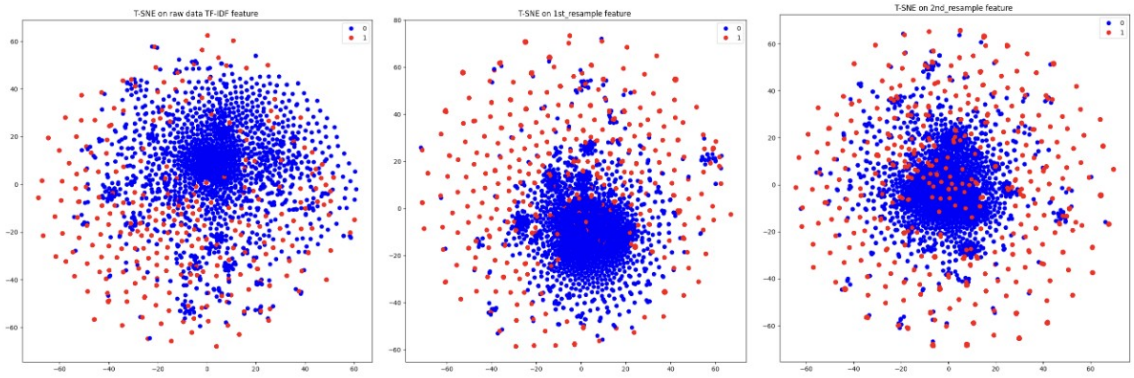
Figure 13. Prediction performance before and after data cleaning

4.3 Dataset resampling and prediction accuracy

Second, we analyzed the impact of resampling, employing two oversampling schemes using the SMOTE algorithm. The first one involved oversampling the minority class based on the entire dataset, followed by the split into training and test data (illustrated in Figure 14(a)). The second scheme split the dataset first and then oversampled only the training data (depicted in Figure 14(b)). In the first scheme, due to random splitting, the sample volume for each class was uneven in the training dataset, and the oversampling generated additional samples that were also included in the test dataset. In the second scheme, oversampling occured after the training and test dataset splitting, resulting in a balanced distribution of generated samples within the training dataset exclusively.

Figure 14. Different sample volumes using SMOTE indifferent schemes

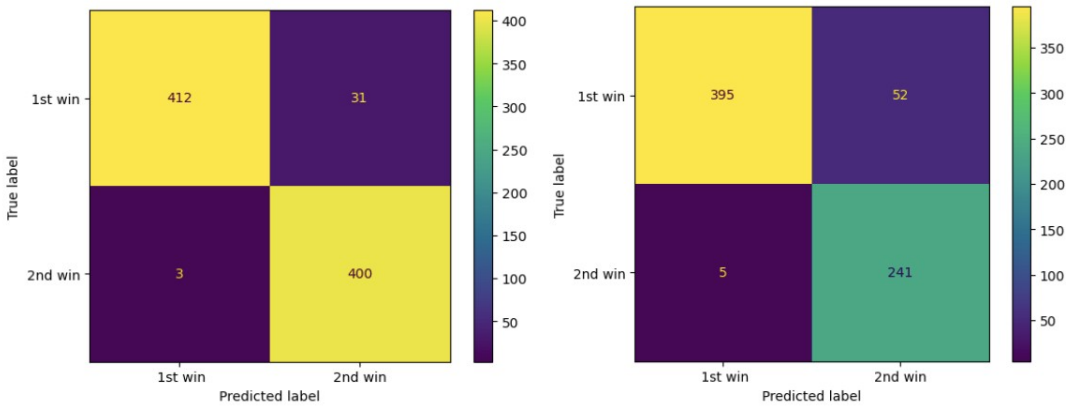
After vectorization, each case was represented by a vector length of 14,713. It is challenging to understand, explore, and analyze to find the pattern hidden in the data due to its high dimensionality. To better visualise the resampling results, t-SNE, a method for creating two-dimensional "maps" of high-dimensional features (22) was employed to display the new samples' distribution, which is shown in Figure 15.



(a) t-SNE on original data (b) t-SNE on 1st scheme (c) t-SNE on 2nd scheme

Figure 15. t-SNE on TF-IDF features

Figure 15 (a) shows that the red dots (label =1) and blue dots (label=0) are not randomly distributed. Most blue dots gather in the center, and some blue dots and most red dots are dispersed outward. The distribution in Figure 15(b) for the first resampling scheme follows similar patterns. However, Figure 15(c) depicts a scenario where the generated red dots are scattered in the center and mixed with blue dots, presenting a more challenging classification task. To further evaluate the impact, Figure 16 displays the prediction results using a confusion matrix.



(a) Confusion matrix using 1st resampling scheme (b)Confusion matrix using 2nd resampling scheme

Figure 16. Confusion matrices

The prediction accuracy of the first resampling scheme was 94.09%, while that for the second resampling scheme was 91.77% . From the confusion matrix in Figure 16, the 1st resampling method's prediction error rate of the "1st party win" (label=0) was 7%, and 11.6% for the 2nd resampling method. Compared with the results without any resampling, which showed a 10.3% error rate, the 1st resampling method was more efficient. For the "2nd party win" (label =1), we observed that the first resampling method introduced additional minor samples, as shown in Figure 16(a). Despite this increase, the prediction error rate remained low at about 0.7%. Conversely, for the 2nd resampling scheme, no oversampling samples were injected into the test dataset. However, the associated prediction error rate was 2%, which was higher than its counterpart without resampling. The first resampling method displays a higher accuracy compared to the second in the confusion matrices (Figure 16).

4.4 Avoiding overfitting and increasing prediction accuracy

To avoid overfitting, we optimized the neural network in two ways: the first was to simplify the networks based on the baseline model, and the second was to add the dropout in the network. The results are presented in Table 2.

Table 2. Performance on avoiding overfitting using the 1st resampling scheme

Kernel number	ACC	Dropout (0.5)	Layers reduced	ACC	Dropout (0.2)
(256,128,64)	94.09%	**96.10%↑**	(256,64)	**94.33%↑**	**94.92↑↑**
(128,64,32)	**94.56%↑**	**95.74%↑↑**	(128,32)	**95.04%↑**	**94.44%↑**
(64,32,16)	**95.63%↑**	**95.98%↑↑**	(64,16)	93.50%↓	93.38%↓

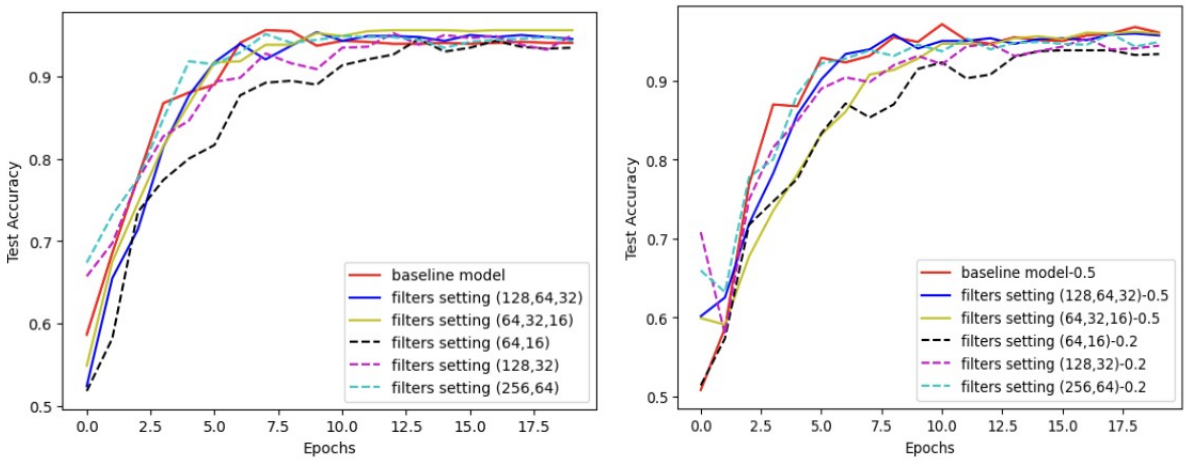


Figure 17. Validation histories in various settings (1st resampling scheme)

Table 3. Performance on avoiding overfitting using the 2nd resampling scheme

Kernel number	ACC	Dropout (0.5)	Layers reduced	ACC	Dropout (0.2)
(256,128,64)	**91.77**	**94.08%↑↑**	(256,64)	**94.23%↑**	**94.37%↑↑**
(128,64,32)	**91.92%↑**	**94.23%↑↑**	(128,32)	**93.36%↑**	**94.95%↑↑**
(64,32,16)	90.62%	**93.22%↑↑**	(64,16)	**93.50%↑**	91.49%

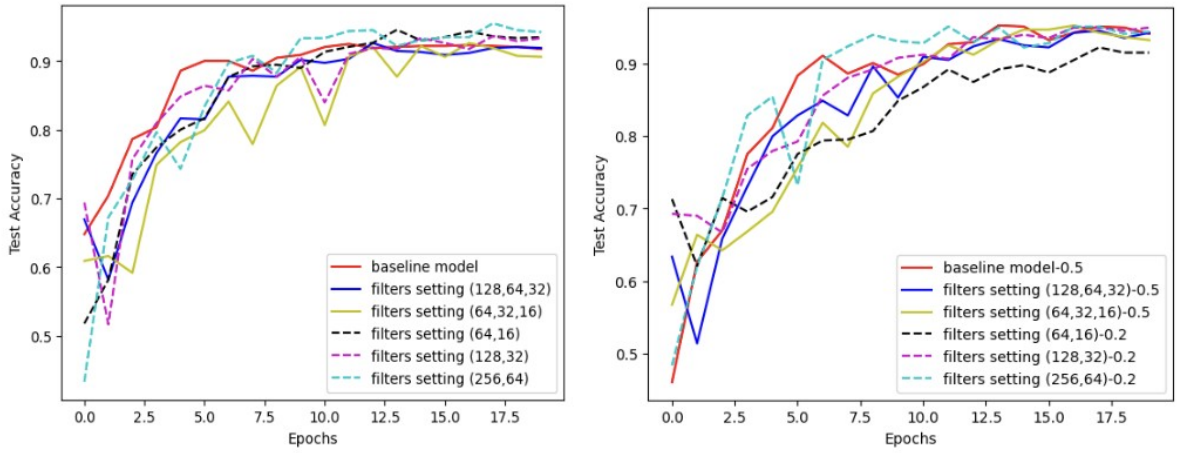


Figure 18. Validation histories in various settings (2nd resampling scheme)

In Tables 2 and 3, we used different colors and arrow directions to represent the comparison results with the baseline. For example, in Table 2, the ACC of the baseline model was 94.09%, and when the dropout (0.5) was employed, the ACC increased to 96.10%; the number was presented in purple color, and with an upward arrow, which in turn meant that the dropout was efficient, and when compared with the baseline model, the ACC was increased. For the third row, when the Kernel number was reduced to (128, 64, 32), the ACC was calculated to be 94.56%, the number was presented in red with an arrow. This provided evidence that simplifying the network by reducing the Kernel number was efficient, and the ACC was increased compared with the baseline model. When the dropout (0.5) was added, the ACC was calculated to be 95.74% with two upward arrows, which implied that the dropout boosted the performance compared with its counterpart, and was higher than the baseline model. In this table, the ACC was 94.44% represented with a green colored number with one upward arrow. This was higher than the ACC of the baseline model but did not boost performance when the dropout (0.2) was introduced in that network setting. In addition, when the network setting was (64, 16), the ACC with dropout (0.2) as well as the ACC without dropout were found to be lower than the baseline. This implied

that there was a decrase in performance when the neural networks were oversimplified.

Furthermore, Figures 17 and 18 display the validation histories. A plateau resulted before 20 epochs in every situation. Specifically, in Figure 17, when dropout was added, it converged close to the 10th epoch which was later than convergence without dropout at the 8th epoch. This phenomenon is more obvious in Figure 18, where convergence without dropout occurs at the 10^{th} epoch, but does not converge until the 17th epoch under the situation with dropout.

In general, both network simplification and dropout contributed to improving model performance. However, oversimplifying was detrimental to the model accuracy. Moreover, the dropout did delay the model convergence, but was in an acceptable range.

4.5 Examples

Based on the analysis and experiments above, we designed the cleaning overflow in figure 7, followed by vectorization using TF-IDF; transformation of the cleaned text to numerical numbers, and finally arrived at the model with hyperparameters as shown in Table 1. We present the 3 cases below as examples. The workflow schematic is shown in Figure 4.

4.5.1 Case 1

Input : In 1970, Jane Roe (a fictional name used in court documents to protect the plaintiff's identity) filed a lawsuit against Henry Wade, the district attorney of Dallas County, Texas, where she resided, challenging a Texas law making abortion illegal except by a doctor's orders to save a woman's life. In her lawsuit, Roe alleged that the state laws were unconstitutionally vague and abridged her right of personal privacy, protected by the First, Fourth, Fifth, Ninth, and Fourteenth Amendments.

Data cleaning: jane roe fictional name used court document protect plaintiff identity filed lawsuit henry wade district attorney dallas county texas resided challenging texas law making abortion illegal except doctor order save woman life lawsuit roe alleged state law unconstitutionally vague abridged right personal privacy protected first fourth fifth ninth fourteenth amendment

Numerical features vector (just show the first 50 as example):
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.15071152 0. 0. 0. 0.21178926]

First party: Jane Roe
Second Party: Henry Wade
System output in probability: [9.9997777e-01, 2.2169283e-05]
System final output: 0
The Winner Party : Jane Roe

4.5.2 Case 2

Input : In 2006, the City of San Antonio, Texas, filed a class - action lawsuit against various online travel companies (OTCs), such as Hotels. com, Hotwire, Orbitz, and Travelocity, alleging that the high fees those companies charged constitute the " cost of occupancy " and therefore are subject to municipal hotel tax ordinances. After extensive litigation, the you. S. Court of Appeals for the Fifth Circuit ruled in favor of the OTCs, reasoning that a hotel occupancy tax applied equally to the discounted room rate paid by the OTC to the hotel. Toward the end of litigation, the OTCs moved for " an order entering Final Judgment in favor of the OTCs, releasing all supersedeas bonds, and awarding costs to the OTCs as the prevailing parties. " The OTCs ' proposed order stated that " costs shall be taxed against the Cities in

favor of the OTCs pursuant to 28 you. S. C. § 53, Fed. R. Civ. P. 54, and Fed. R. App. P. 39. " San Antonio did not object, so the district court entered the OTC ’ s proposed order. Then, OTCs filed a bill of costs in the district court seeking over $ 2. 3 million, which included over $ 2 million for “ post - judgment interest ” and “ premiums paid for the appeal bonds. ” San Antonio objected and asked the district Court to refuse to tax, but to substantially reduce, the appeal bond premiums sought by the OTCs. The district court concluded that it lacked the discretion to reduce taxation of the bond premiums. The Fifth Circuit affirmed, despite that every other circuit confronting the question has held the opposite.

Data cleaning: city san antonio texas filed class action lawsuit various online travel company otcs hotel com hotwire orbitz travelocity alleging high fee company charged constitute cost occupancy therefore subject municipal hotel tax ordinance extensive litigation court appeal fifth circuit ruled favor otcs reasoning hotel occupancy tax applied equally discounted room rate paid otc hotel toward end litigation otcs moved order entering final judgment favor otcs releasing supersedeas bond awarding cost otcs prevailing party otcs proposed order stated cost shall taxed city favor otcs pursuant c fed civ fed app san antonio not object district court entered otc proposed order otcs filed bill cost district court seeking million included million post judgment interest premium paid appeal bond san antonio objected asked district court refuse tax substantially reduce appeal bond premium sought otcs district court concluded lacked discretion reduce taxation bond premium fifth circuit affirmed despite every circuit confronting question held opposite

Numerical features vector (just show the first 50 as example):

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

The first party : City of San Antonio, Texas, On Behalf of Itself and All Other Similarly Situated Texas Municipalities
The second Party: Hotels.com, L.P., et al.
System output in probability:[0.0576881, 0.9423118]
System final output: 1
The Winner Party : Hotels.com

4.5.3 Case 3

Input : Joan Stanley had three children with Peter Stanley. The Stanleys never married, but lived together off and on for 18 years. When Joan died, the State of Illinois took the children. Under Illinois law, unwed fathers were presumed unfit parents regardless of their actual fitness and their children became wards of the state. Peter appealed the decision, arguing that the Illinois law violated the Equal Protection Clause of the Fourteenth Amendment because unwed mothers were not deprived of their children without a showing that they were actually unfit parents. The Illinois Supreme Court rejected Stanley’s Equal Protection claim, holding that his actual fitness as a parent was irrelevant because he and the children’s mother were unmarried.

Data cleaning: joan stanley three child peter stanley stanley never married lived together year joan died state illinois took child illinois law unwed father presumed unfit parent regardless actual fitness child became ward state peter appealed decision arguing illinois law violated equal protection clause fourteenth amendment unwed mother not deprived child without showing actually unfit

parent illinois supreme court rejected stanley equal protection claim holding actual fitness parent irrelevant child mother unmarried

Numerical features vector (just show the first 50 as example):
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

The first party : Peter Stanley, Sr.
The second party:Illinois
System output in probability:[0.99655735, 0.0034425]
System final output: 0
The Winner Party : Peter Stanley, Sr.

5 Limitations

Our model has various potential applications, including improving the efficiency of legal experts by assisting them in processing facts and ensuring accurate predictions. It can also provide real-time legal consultations to the public, helping them understand how certain behaviors align with predicted outcomes. In addition, the model can serve as an educational tool to enhance public knowledge and access to law, judgments, and justice. However, currently, our model has been trained on data from US Supreme Court. As a result, when the test data is from another legal system, it may not accurately predict judicial outcomes due to the inherent differences in legal principles and practices.

6 Conclusion

This paper explored the use of AI to model an assistant system to automatically predict the judgement of court cases based on the description of the facts. We selected the dataset from the US Supreme Court and implemented the 1D convolutional network for predicting judgement outcomes. We incorporated data resampling and cleaning modules to address class imbalances and data noise. In addition, network simplification and dropout methods were adopted to avoid overfitting during the training. Our AI model, that we termed the auto-judgement system, demonstrated a > 95% accuracy when predicting the result of a subset of cases from the US Supreme Court. In future work, we will shift our focus toward improving feature representation of texts, since it represents a critical step for enhancing prediction accuracy.

References

1. R. A. Oppel Jr, J. K. Patel, (2019). One lawyer, 194 felony cases, and no time, New York Times 31. https://www.nytimes.com/interactive/2019/01/31/us/public-defender-case-loads.html

2. L. Garcia-Navarro, P. MOURA, (2014). Brazil: The land of many lawyers and very slow justice. https://www.npr.org/sections/parallels/2014/11/05/359830235/brazil-the-land-of-many-lawyers-and-very-slow-justice

3. Cui, J., Shen, X., Nie, F., Wang, Z., Wang, J., Chen, Y., (2022). A survey on legal judgement prediction: Datasets, metrics, models, and challenges. *arXiv preprint arXiv:2204.04859*. https://arxiv.org/abs/2204.04859

4. About the Supreme Court. (n.d.). United States Courts. https://www.uscourts.gov/about-federal-courts/educational-resources/about-educational-outreach/activity-resources/about

5. Gross, S. R., O'Brien, B., Hu, C., Kennedy, E. H. (2014). Rate of false conviction of criminal defendants who are sentenced to death. *Proceedings of the National Academy of Sciences*, *111*(20), 7230–7235. https://doi.org/10.1073/pnas.1306417111

6. Alsayed, S. Syed, M. Alali, S. Patel, H. Bodala, (2021). Justice: A benchmark dataset for supreme court's judgment prediction, preprint arXiv:2112.03414. https://arxiv.org/abs/2112.03414

7. Contributors to Wikimedia projects. *Common law*. Wikipedia. https://en.wikipedia.org/wiki/Common_law

8. Vajjala, S., Majumder, B., Gupta, A., Surana, H. (2020). *Practical natural language processing: A comprehensive guide to building real-world NLP systems*. O'Reilly Media. https://www.practicalnlp.ai/

9. R. C. Lawlor, (1963). What computers can do: Analysis and prediction of judicial decisions, American Bar Association Journal 337–344. https://www.jstor.org/stable/25722338

10. Semo, G., Bernsohn, D., Hagag, B., Hayat, G., Niklaus, J. (2022). ClassActionPrediction: A challenging benchmark for legal judgment prediction of class action cases in the US. *Proceedings of the Natural Legal Language Processing Workshop 2022*. http://dx.doi.org/10.18653/v1/2022.nllp-1.3

11. Sharma, S. K., Shandilya, R., Sharma, S. (2022). Predicting Indian Supreme Court judgments, decisions, or appeals. *Statute Law Review*, *44*(1). https://doi.org/10.1093/slr/hmac006

12. Hwang, W., Lee, D., Cho, K., Lee, H., Seo, M. (2022). "A multi-task benchmark for Korean legal language understanding and judgement prediction." *Advances in Neural Information Processing Systems* 35: 32537-32551. https://arxiv.org/abs/2206.05224

13. Niklaus, J., Chalkidis, I., Stürmer, M. (2021). Swiss-Judgment-Prediction: A multilingual legal judgment prediction benchmark. *Proceedings of the Natural Legal Language Processing Workshop 2021*. http://dx.doi.org/10.18653/v1/2021.nllp-1.3

14. Dealing with imbalanced datasets in machine learning: Techniques and best practices. *Sole*. https://www.blog.trainindata.com/machine-learning-with-imbalanced-data/

15. Al Sharou, K., Li, Z., Specia, L. (2021). Towards a better understanding of noise in natural language processing. *Proceedings of the Conference Recent Advances in Natural*

Language Processing - Deep Learning for Natural Language Processing Methods and Applications. http://dx.doi.org/10.26615/978-954-452-072-4_007

16. Raghunathan, D. (2020, June 9). NLP in Python-Data cleaning. *Towards Data Science*. https://towardsdatascience.com/nlp-in-python-data-cleaning-6313a404a470

17. Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

18. Goyal, P., Pandey, S., Jain, K. (2018). *Deep learning for natural language processing: Creating neural networks with python*. Apress. https://link.springer.com/book/10.1007/978-1-4842-3685-7

19. What is Overfitting? (n.d.). IBM. https://www.ibm.com/topics/overfitting

20. Sanjar, K., Rehman, A., Paul, A., JeongHong, K. (2020, December 18). Weight dropout for preventing neural networks from overfitting. *2020 8th International Conference on Orange Technology (ICOT)*. http://dx.doi.org/10.1109/icot51877.2020.9468799

21. *Google colaboratory*. (n.d.). https://colab.research.google.com/?utm_source=scs-index https://colab.research.google.com/github/alzayats/Google_Colab/blob/master/4_4_overfitting_and_underfitting.ipynb#scrollTo=kB1lu4iqRbQr

22. Saurabh.jaju2. (2017, January 22). *Guide to t-SNE machine learning algorithm implemented in R & Python*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/