Development and application of intelligent judicial trial assistance system based on generative artificial intelligence and machine learning technology

Yaohao Lian[1,2,*], Yining Yang[1,3]

[1]Hebei University of Economics and Business, Hebei, 050000, China

[2]2052653670@qq.com
[3]3195587298@qq.com
[*]corresponding author

Abstract. The intelligent judicial trial assistance system is an intelligent auxiliary judicial trial tool based on generative artificial intelligence and machine learning technology. It can achieve auxiliary functions such as case upload, trial result prediction, and legal document generation through automated, digital, and informational means. It utilizes technologies such as Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM) to train generative models for generating trial recommendations and assisting in decision-making. Through in-depth research on the working principles and optimization paths of the intelligent judicial trial assistance system, the best applicable path for the system in judicial trial practice is explored. By establishing optimized models, risks and biases are minimized to the greatest extent.

Keywords: Intelligent judiciary, Machine learning, Generative artificial intelligence, Deep neural networks, Model training.

1. Introduction

With the accelerating process of information society, facing the high cost of information acquisition, low efficiency, uncertainty of trial results, and the issue of insufficient judicial resources due to "high case volume and few personnel" in the traditional judicial trial process [1], this project aims to promote the construction of informatization in the judicial field through the development of a well-functioning intelligent judicial trial assistance system. At the same time, it accelerates the construction process of the innovative "AI + Legal Studies" model, providing more convenient and fair judicial services to all sectors of society. It offers judges an interpretable decision-making process, increasing the scientific and legal integrity of judicial trials. Meanwhile, it optimizes resource allocation, enhances the efficiency of judicial resource utilization, alleviates the situation of "high case volume and few personnel" faced by China's judiciary, and improves the work efficiency and public service level of judicial organs [3].

2. Implementation Path Planning

2.1. Functional Path Planning

（1）User Registration and Login:

Users (such as judges, legal personnel, etc.) need to register and log in to the system to ensure the independence and security of each user. This system primarily uses Python's Flask framework and requests library to implement user registration and login functionality based on HTTP POST.

（2）Case Upload:

Users need to upload or input basic information about a case, such as case type, legal provisions, involved parties' information, evidence materials, etc. This data is collected and organized by the system as a basis for subsequent analysis, mainly through Flask server, SQLAlchemy, Spacy, and NLTK technologies to implement uploading, processing, extracting, and saving case information to the database.

（3）Data Processing and Preprocessing:

The system processes and preprocesses the case data uploaded by users, such as data cleaning, feature extraction, data encoding, etc., as preparation for subsequent machine learning model training. Using pandas and scikit-learn libraries to read, load, process, and prepare case data, loading data into pandas DataFrame from SQLAlchemy database, then splitting, cleaning, and processing the loaded case data. Continuing with feature selection and preprocessing, separating features and target variables from the dataset, encoding categorical variables, extracting useful characteristics from text columns, and finally splitting the dataset into training and test sets.

（4）Prediction Model Generation:

The preprocessed data can be used to build and train prediction models using XGboost and scikit-learn, to implement functions such as case outcome prediction, case duration prediction, key influencing factors identification, and case trend prediction. Through the implementation of prediction functions, users can have a more reasonable expectation of the case judgment. Case outcome prediction, case duration prediction, and key influencing factors identification can be well implemented using the XGboost model, while case trend prediction requires the use of time series model ARIMA, which necessitates preprocessed data containing time information and being sorted by time.

（5）Generation of Trial Suggestions and Assistance in Decision Making:

By training the model with a large amount of historical case data, conducting model learning, and using this as the basis for generating trial suggestions and assisting in decision-making. Utilizing Seq2Seq models, attention mechanisms, RNN, and LSTM for the construction of generative models, first creating two inputs, one for the encoder and one for the decoder, using embedding layers for vector embedding of these two inputs. Then, creating an LSTM encoder to receive the embedded encoder input and generate two hidden states. Next, using an LSTM decoder, taking the encoder's hidden states as initial states, receiving the embedded decoder input, and decoding new outputs. Following this, using an attention layer, which receives outputs from both encoder and decoder, by calculating attention weights, to decide which part of the input sequence to focus on more, producing attention results. Finally, connecting the decoder's output and attention results, passing them through a fully connected layer to output the desired result and define the model. To train the model, data preparation is needed, including input data of cases (such as case details) and relevant target data (such as expected judgment outcomes). Using a suitable dataset to train the attention mechanism, during the training process, the model will attempt to learn how to produce the correct judgment outcomes from input cases, adjusting model parameters to achieve this goal. Once the model is trained, given new case inputs, the model will generate judgment suggestions based on previous training, providing references for generating trial suggestions and assisting in decision-making for users.

（6）System Learning and Optimization:

To optimize the system based on user feedback and actual case outcomes, a cyclic feedback system is needed. This system collects actual case outcomes and user feedback, using this information as new data to train and optimize the prediction models. By creating a Deep Neural Network (DNN) and training and optimizing it using the Deep Q-Learning (DQN) algorithm, the network initially predicts the maximum potential reward of the next state and updates the model's valuation of the current state and action accordingly. If we reach the end state, then the reward is what has already been obtained.

Otherwise, we add a discounted maximum potential reward as our target value. Then the network optimizes by matching predictions and targets.

（7）Trial Conclusion and Case Archiving:

The system archives all case materials, prediction results, trial processes, and final outcomes for future queries and learning. Creating an SQLite database and establishing a case table to save case information, each record contains the case ID, name, judgment, etc., and a status bit to record whether the case has been archived. Then, inserting a case record, marking a case as archived status. Following this, using the Whoosh module to create a full-text search index, then adding case information to this index for full-text searching.

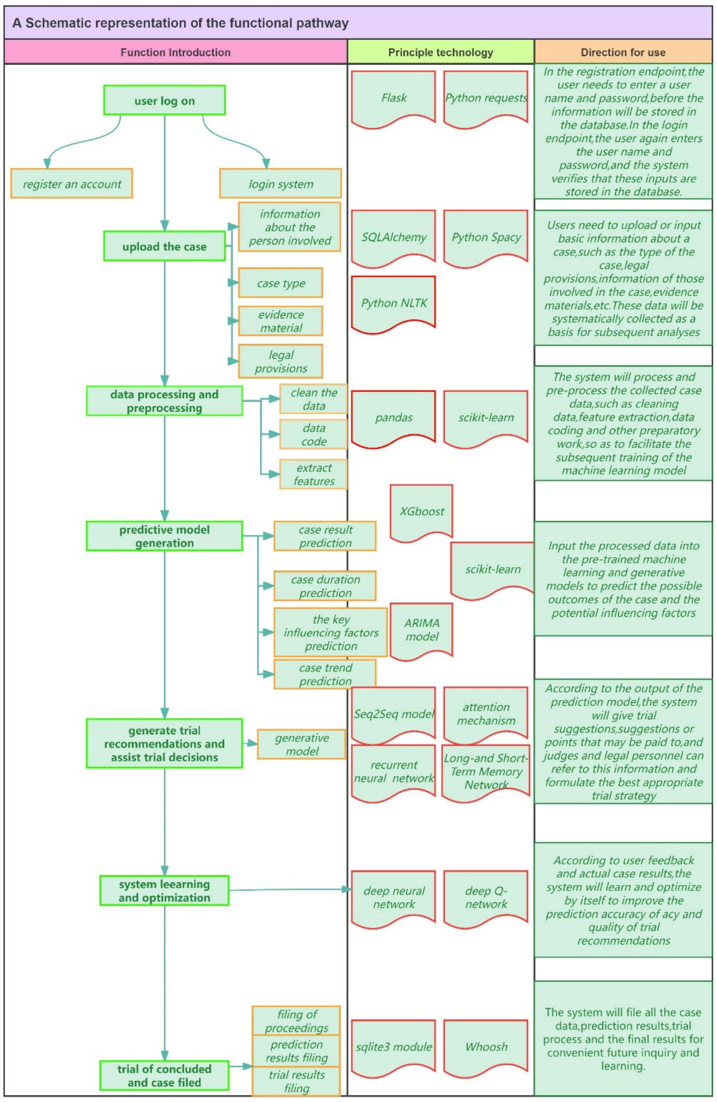
A Schematic representation of the functional pathway
Function Introduction
Principle technology
Direction for use
user log on
register an account
login system
upload the case
information about the person involved
case type
evidence material
legal provisions
data processing and preprocessing
clean the data
data code
extract features
predictive model generation
case result prediction
case duration prediction
the key influencing factors prediction
case trend prediction
generate trial recommendations and assist trial decisions
generative model
system leearning and optimization
trial of concluded and case filed
filing of proceedings
prediction results filing
trial results filing
Flask
Python requests
SQLAlchemy
Python Spacy
Python NLTK
pandas
scikit-learn
XGboost
scikit-learn
ARIMA model
Seq2Seq model
attention mechanism
recurrent neural network
Long-and Short-Term Memory Network
deep neural network
deep Q-network
sqlite3 module
Whoosh
In the registration endpoint,the user needs to enter a user name and password,before the information will be stored in the database.In the login endpoint,the user again enters the user name and password,and the system verifies that these inputs are stored in the database.
Users need to upload or input basic information about a case,such as the type of the case,legal provisions,information of those involved in the case,evidence materials,etc.These data will be systematically collected as a basis for subsequent analyses
The system will process and pre-process the collected case data,such as cleaning data,feature extraction,data coding and other preparatory work,so as to facilitate the subsequent training of the machine learning model
Input the processed data into the pre-trained machine learning and generative models to predict the possible outcomes of the case and the potential influencing factors
According to the output of the prediction model,the system will give trial suggestions,suggestions or points that may be paid to,and judges and legal personnel can refer to this information and formulate the best appropriate trial strategy
According to user feedback and actual case results,the system will learn and optimize by itself to improve the prediction accuracy of acy and quality of trial recommendations
The system will file all the case data,prediction results,trial process and the final results for convenient future inquiry and learning.

Figure 1. Functional Path Diagram

2.2. Process Planning

Process One: New users need to register an account through the user interface. In the registration endpoint, users are required to enter a username and password, then this information will be stored in the database. At the login endpoint, users enter their username and password again, and the system will verify whether these inputs match the information stored in the database. After logging in, users can upload cases through the user interface.

Registration for new users:

```
import requests
import json
def register_user(username, password):
url = 'http://your_website.com/register'
headers = {'Content-Type': 'application/json',}
payload = {'username': username,'password': password, }
response = requests.post(url, headers=headers, data=json.dumps(payload))
if response.status_code == 200:
print('user registration succeeded!')
else:
print('user registration failure!')
register_user('MyUsername', 'MyPassword')
```

User login:

```
import requests
import json
def user_login(username, password):
url = 'http://your_website.com/login'
headers = {'Content-Type': 'application/json',}
payload = {'username': username, 'password': password,}
response = requests.post(url, headers=headers, data=json.dumps(payload))
if response.status_code == 200:
print('user login successful!')
else:
print('user login failure!')
user_login('MyUsername', 'MyPassword')
```

Process Two: Users select the type of case they are uploading (criminal, civil, administrative), information about the people involved, evidence materials, legal provisions, and other case information. The system will receive and save the case information uploaded by users; read and process the text data in the file; perform natural language processing; use SQLAlchemy to save the processing results to the database.

Initially, set up a simple Flask server to receive uploaded case information:

```
from flask import Flask, request
from werkzeug.utils import secure_filename
import os
app = Flask(__name__)
@app.route('/upload', methods=['POST'])
def upload_file():
if request.method == 'POST':
f = request.files['file']
filename = secure_filename(f.filename)
f.save(os.path.join("uploads", filename))
process_file(os.path.join("uploads", filename))
return 'File uploaded and processed successfully'
if __name__ == "__main__":
app.run(debug=True)
```

Then process the case information, read file content, and perform natural language processing and saving to the database operations:

```
from sqlalchemy import create_engine, MetaData, Table
from sqlalchemy.orm import sessionmaker
import spacy
from nltk.corpus import stopwords
nlp = spacy.load("en_core_web_sm")
stop_words = set(stopwords.words('english'))
def process_file(filepath):
with open(filepath, 'r') as f:
text = f.read()
doc = nlp(text)
nouns = [chunk.text for chunk in doc.noun_chunks]
verbs = [token.lemma_ for token in doc if token.pos_ == "VERB"]
nouns = [word for word in nouns if word not in stop_words]
verbs = [word for word in verbs if word not in stop_words]
engine = create_engine('sqlite:///cases.db')
metadata = MetaData()
cases = Table('cases', metadata, autoload_with=engine)
Session = sessionmaker(bind=engine)
session = Session()
new_case = cases.insert().values(file_path=filepath, nouns=nouns, verbs=verbs)
session.add(new_case)
session.commit()
```

Process Three: After the system cleans and preprocesses the uploaded case data, it imports them into the predictive and generative models. The system can generate predictions on the case outcomes, duration, influencing factors, and trends based on the trained models, and generate trial suggestions and assistance in decision-making tailored to the uploaded cases.

Process Four: The system archives the adjudicated case materials, trial process, and trial results, and creates retrieval methods for later review and learning.

Perform case closure operations, setting the case to archived status:

```
import sqlite3
conn = sqlite3.connect('case.db')
c = conn.cursor()
sql_create = '''CREATE TABLE CASE(ID INT PRIMARY KEY NOT NULL,NAME
TEXT NOT NULL,VERDICT TEXT NOT NULL,DECISION TEXT,ARCHIVED INT);'''
c.execute(sql_create)
sql_insert = '''INSERT INTO CASE(ID,NAME,VERDICT,DECISION,ARCHIVED)
VALUES (1, 'Case A', 'Guilty', '10 years in prison', 0)'''
c.execute(sql_insert)
conn.commit()
sql_archive = '''UPDATE CASE set ARCHIVED = 1 where ID = 1'''
c.execute(sql_archive)
conn.commit()
```

Create a method for retrieving historical cases:

```
from whoosh.index import create_in
from whoosh.fields import *
from whoosh.qparser import QueryParser
schema = Schema(id=NUMERIC(stored=True), content=TEXT)
index = create_in("whoosh_index", schema)
writer = index.writer()
```

```
writer.add_document(id=1, content='Case A Guilty 10 years in prison')
writer.commit()
searcher = index.searcher()
parser = QueryParser("content", index.schema)
myquery = parser.parse("Guilty")
results = searcher.search(myquery)for hit in results:
print(hit["id"])
```

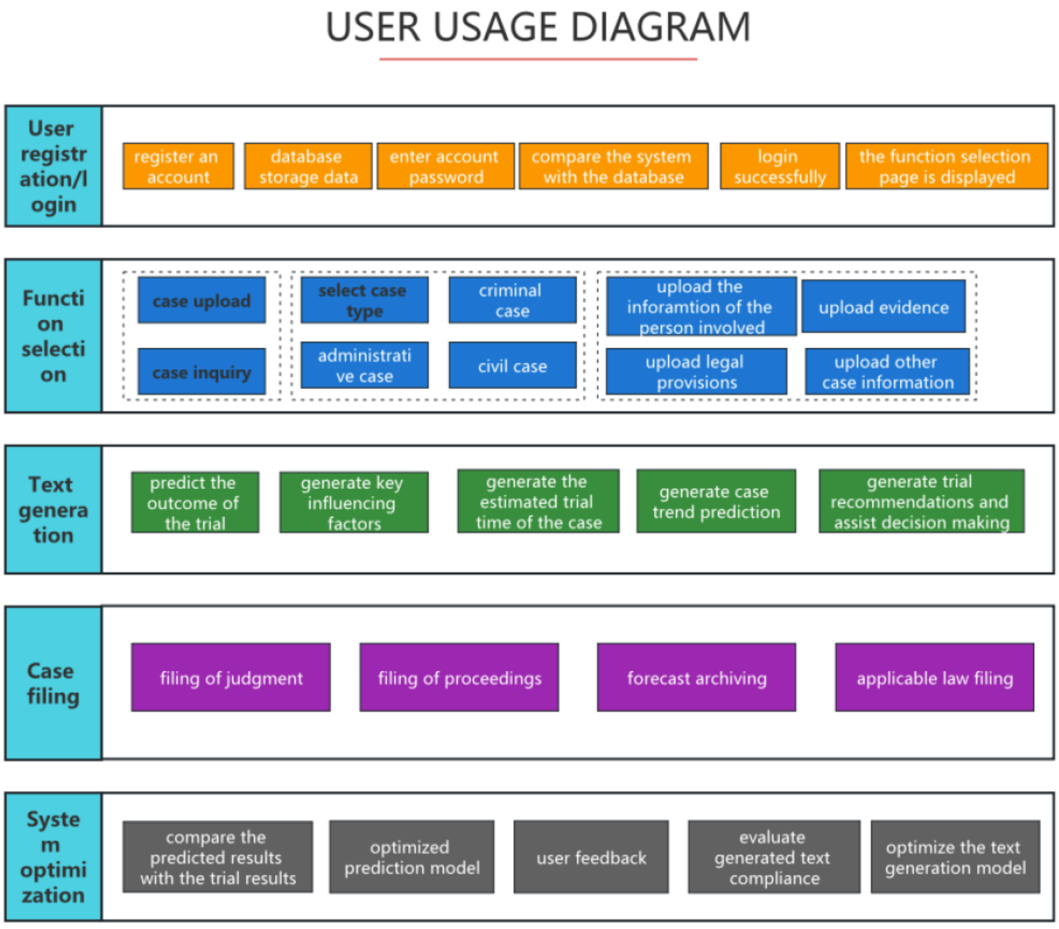
USER USAGE DIAGRAM
User registration/login
register an account
database storage data
enter account password
compare the system with the database
login successfully
the function selection page is displayed
Function selection
case upload
case inquiry
select case type
criminal case
administrative case
civil case
upload the inforamtion of the person involved
upload evidence
upload legal provisions
upload other case information
Text generation
predict the outcome of the trial
generate key influencing factors
generate the estimated trial time of the case
generate case trend prediction
generate trial recommendations and assist decision making
Case filing
filing of judgment
filing of proceedings
forecast archiving
applicable law filing
System optimization
compare the predicted results with the trial results
optimized prediction model
user feedback
evaluate generated text compliance
optimize the text generation model

Figure 2. User Usage Diagram

3. Prospects and Challenges

The application of intelligent judicial trial assistance systems in the field of judicial practice can help judicial personnel automatically process legal documents and cases, reduce manual workload, assist judges in making more rational judgments, and improve judicial efficiency. It can also be used for public legal education, enabling the public to better understand and apply the law [1]. However, the practical application of the system also faces many challenges. Legal cases often involve the private information of multiple parties, and how to conduct efficient machine learning while meeting data protection and privacy regulations is a challenge. Moreover, training accurate machine learning models requires a vast amount of high-quality data. The legal field has strict regulations and standards, and ensuring the system's output complies with these regulations and standards is also a major challenge [5]. Despite the increasingly powerful decision-making capabilities of machine learning models, their interpretability often remains poor, which could affect user trust and acceptance. In summary, while the intelligent judicial trial assistance system presents significant prospects, it also faces numerous challenges. Overcoming these challenges to explore a truly suitable application path is necessary to unlock the system's potential value in the judiciary [2].

4. Conclusion

This system is an intelligent judicial trial assistance system that combines generative artificial intelligence and machine learning technologies. It aids professionals in making more accurate judgments and decisions by analyzing and learning from a vast array of legal cases and regulations. Its design concept includes needs analysis, model selection, data preparation, model training and validation, system design and testing, and finally deployment and optimization [4]. The emergence of this system signifies the arrival of an era of intelligence, expected to significantly improve the work efficiency of legal professionals. Additionally, with the continuous increase in data, the system's judgment capability and accuracy are expected to improve over the long term, making it a powerful legal assistance tool. However, from a practical and application perspective, the system will also face some challenges, such as how to protect data privacy, improve data quality, solve technical complexity, comply with regulatory requirements, enhance system interpretability and trust, avoid erroneous decisions, and ensure the system's continuous update and maintenance. Therefore, overcoming these issues through technological improvements and strict management is necessary to realize the system's maximum potential [6].

References

[1] Xu, H., & Li, J. Q. (2023). The possibilities and boundaries of generative artificial intelligence in assisting judicial adjudication. Journal of Taiyuan University of Technology (Social Science Edition), 41(06), 24-32.

[2] Guo, W. J. (2023). The dual dilemma of factual determination in the application of artificial intelligence in judiciary and the path to relief. Journal of Kunming University of Science and Technology (Social Science Edition), 23(05), 18-25. https://doi.org/10.16112/j.cnki.53-1160/c.2023.05.213

[3] Zheng, X. (2023). The application of generative artificial intelligence in judiciary: Prospects, risks, and regulations. Chinese Journal of Applied Jurisprudence, (04), 81-93.

[4] Zhao, G. (2023). Research on the choice of game strategy and applicability evaluation of artificial intelligence application modes in judicial adjudication [Doctoral dissertation]. Shenyang University of Technology. https://doi.org/10.27322/d.cnki.gsgyu.2023.000001

[5] Ni, H. C. (2023). The prevention mechanism of wrongful criminal cases under the perspective of intelligent judiciary [Doctoral dissertation]. Zhejiang Gongshang University. https://doi.org/10.27462/d.cnki.ghzhc.2023.000086

[6] He, Y. J., & Lei, Z. W. (2022). Challenges and responses: Judicial adjudication in the era of artificial intelligence. Journal of Guizhou Minzu University (Philosophy and Social Sciences Edition), (01), 178-190.