IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

KANOONSAMPARKA-LEGAL CONNECT AND CASE ANALYSIS PLATFORM

Shilpa P Madhavanavar[1], Bumika N Surpur[2], Keerti P Satti[3], Poorvi K Muttinpendimath[4]

Assistant Professor, Department of Computer Science and Engineering, B.V.V. Sangha's Basaveshwar , Engineering College, Bagalkot[1]

Students, Department of Computer Science and Engineering, B.V.V. Sangha's Basaveshwar , Engineering College, Bagalkot[2,3,4]

Abstract: Citizens seeking legal help often find it difficult to identify an advocate suited to their case, while advocates must manually study FIRs, complaints, petitions and judgments that arrive in different formats before they can understand a case. This paper presents KanoonSamparka, a web-based platform that brings client–advocate matching, structured case requests, document digitisation and AI-assisted preliminary analysis into a single workflow. A TF-IDF and Logistic Regression classifier assigns the client's typed case description to the Civil or Criminal category, which drives advocate matching. Uploaded documents are digitised with EasyOCR and PyPDF and analysed by the Google Gemini API, supported by a retrieval-augmented generation (RAG) module built on Sentence Transformers and FAISS, to produce a document summary, chronological timeline, key points and mind map. The system uses React with Vite, a Flask REST backend and Supabase PostgreSQL. Classification quality, pipeline latency, retrieval quality and backend response times are reported. KanoonSamparka is a decision-support tool that assists, and does not replace, professional legal advice.

I. INTRODUCTION

Access to legal assistance remains a challenge for many individuals. People facing legal issues may find it difficult to identify the nature of their case and select an appropriate advocate. Similarly, advocates often receive legal documents such as FIRs, complaints, petitions and court orders in scanned or digital formats. Examining these records manually requires time and effort, as legal documents may contain lengthy and complex information [2].

Several challenges motivate the development of this work:

(i) difficulty in connecting clients with suitable advocates
(ii) time-consuming examination of legal documents
(iii) challenges in extracting and organising information from different file formats
(iv) the need for an integrated platform that combines advocate discovery, case management and document analysis.

KanoonSamparka aims to address these challenges through a web-based application with separate interfaces for clients and advocates. The main contributions of the proposed system are:

- Client–Advocate Connection**:** Provides facilities for case submission, advocate discovery, request handling and case progress tracking.
- Machine Learning Classification: Uses TF-IDF and Logistic Regression to classify a client's typed case description into exactly two categories: Civil and Criminal. Only the entered description is considered for this prediction.
- Document Processing and Analysis**:** Uses EasyOCR and PyPDF to extract text from uploaded documents. Google Gemini is used to generate summaries, chronological timelines, key points and mind maps.
- Retrieval-Augmented Generation: Uses Sentence Transformers and FAISS to retrieve relevant legal information that can support Gemini during document analysis.

The platform is intended to assist with preliminary legal document examination. Its generated outputs require verification by an advocate and do not replace professional legal advice or judgement.

The remainder of this paper is organised as follows. Section II presents the related work, Section III explains the proposed methodology, Section IV discusses the experimental results and system evaluation, and Section V concludes the paper with possible future enhancements.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511216

II. OBJECTIVES

1. To develop an integrated web-based platform that enables citizens to find advocates based on their legal requirements and manage case requests through a structured process of submission, acceptance, and progress tracking.

2. To process scanned and digital legal documents, such as FIRs, complaints, and court orders, by converting scanned content into searchable text using Optical Character Recognition (OCR) and extracting text directly from digital PDF files.

3. To develop and evaluate a supervised machine learning model using TF-IDF and Logistic Regression to classify cases into two broad legal categories, Civil and Criminal, based exclusively on the case description provided by the client.

4. To integrate a large language model (Gemini) for analysing extracted legal document text and generating preliminary case information, including key facts, important points, document summaries, and a chronological sequence of events to support initial case understanding.

5. To evaluate the performance of the proposed system by examining classification accuracy, document text extraction quality, information retrieval effectiveness, and response time of the major system operations.

III. RELATED WORK

Legal Artificial Intelligence uses Natural Language Processing techniques for applications such as judgment prediction, legal document retrieval and text summarisation, helping reduce routine work for legal professionals [1]. The Indian Legal Documents Corpus (ILDC) provides Supreme Court case documents for research on judgment prediction and explanation, highlighting the challenges involved in processing complex legal records [2]. LEGAL-BERT demonstrates how adapting pre-trained language models to legal terminology can improve their performance on legal language processing tasks [3].

Retrieval-Augmented Generation (RAG) combines information retrieval with text generation, allowing a language model to use relevant external information while producing responses [4]. Sentence-BERT generates sentence embeddings that support semantic similarity comparison [5], while FAISS enables efficient similarity searches across vector collections [6]. TF-IDF assigns weights to words according to their importance within a document collection [8]. When combined with Logistic Regression from Scikit-learn [7], it provides a practical approach to text classification. Existing legal technology platforms, including NyayAI, NyayaGuru and VakilSearch, provide services related to legal information, AI-assisted assistance and advocate discovery. KanoonSamparka focuses on bringing advocate discovery, case request management, OCR-based document processing and Gemini-assisted preliminary analysis together within a unified client–advocate workflow.

IV. PROPOSED METHODOLOGY

A. System Architecture

KanoonSamparka is designed using a separate frontend–backend architecture, as illustrated in Fig. 1. The frontend is developed using React with Vite and provides separate interfaces for clients and advocates. It communicates with the Flask REST API through JSON-based requests over HTTP or HTTPS, depending on the deployment environment. client and advocate interfaces and communicates with the Flask REST API over HTTPS using JSON. The Flask backend is the core computation layer: it handles application logic, runs the ML classifier, receives uploaded documents, calls the OCR, RAG and Gemini components, and performs all database operations. Supabase provides PostgreSQL storage, authentication and file storage. Results, reports and status updates return to the user through the backend and frontend.

The Flask backend acts as the main application processing layer. It manages application logic, executes the Machine Learning classifier, handles document uploads and coordinates the OCR, RAG and Gemini components. Supabase is used for PostgreSQL database management, user authentication and file storage. The processed information, document analysis results, reports and case-status updates are made available to users through the frontend.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

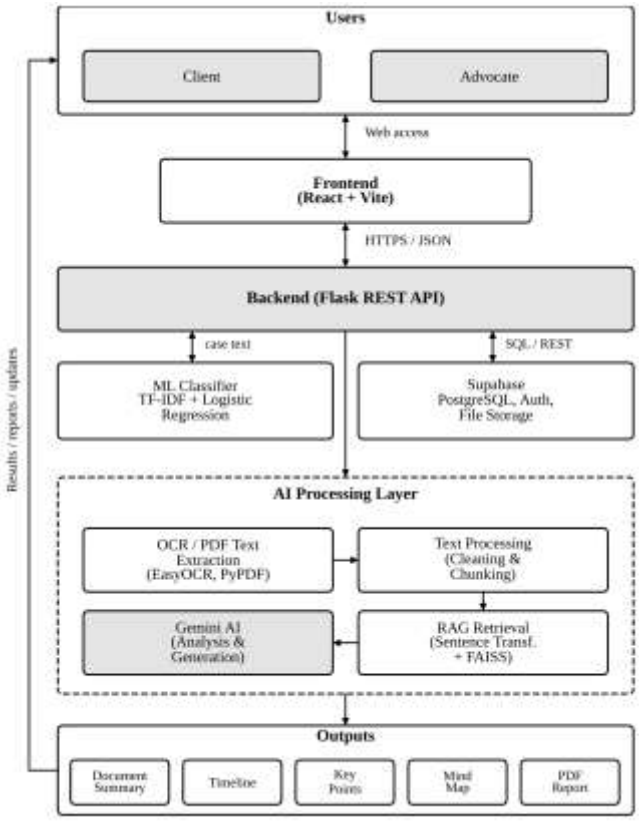


Fig. 1 System architecture of KanoonSamparka

TABLE I TECHNOLOGIES USED AND THEIR PURPOSES

Technology	Purpose
React.js with Vite	Client and advocate dashboards, case forms, document upload
Python with Flask	REST API; application logic, ML, OCR, AI integration, database operations
Supabase	PostgreSQL tables, user authentication, file storage
scikit-learn (TF-IDF + Logistic Regression)	Civil / Criminal classification of the typed case description
EasyOCR and PyPDF	Text extraction from scanned documents and PDF files
Sentence Transformers and FAISS	Text embeddings and similarity-based retrieval for RAG
Google Gemini API	Summary, timeline, key points and mind map generation
ReportLab	Downloadable PDF report for client and advocate

B. User Registration, Login and Dashboards

The application supports two types of users: clients and advocates. Both user types can register and log in through Supabase Authentication. Their profile information and other relevant details are maintained in the database. After successful authentication, users are directed to their respective dashboards based on their roles.

The client dashboard provides facilities for submitting case descriptions, viewing suitable advocates, uploading legal documents and tracking case progress. The advocate dashboard allows advocates to view incoming case requests, accept or reject them, examine assigned cases and update their progress. This separation provides each user type with features relevant to their responsibilities.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

C. ML-Based Case Classification

The Machine Learning module classifies a legal problem based on the description entered by the client. The frontend sends this text to the Flask backend, where it undergoes preprocessing before being converted into numerical features using Term Frequency–Inverse Document Frequency (TF-IDF). This technique assigns importance to words based on their frequency in the input text and their distribution across the training documents.

The extracted TF-IDF features are passed to a Logistic Regression model, which predicts one of exactly two categories: Civil or Criminal (Fig. 2a). The model selects the category with the higher predicted probability.

An important feature of this classification process is that it uses only the case description typed by the client. Uploaded documents and OCR-extracted text are not considered during classification. Google Gemini is also not involved in this prediction process; it is used separately for legal document analysis.

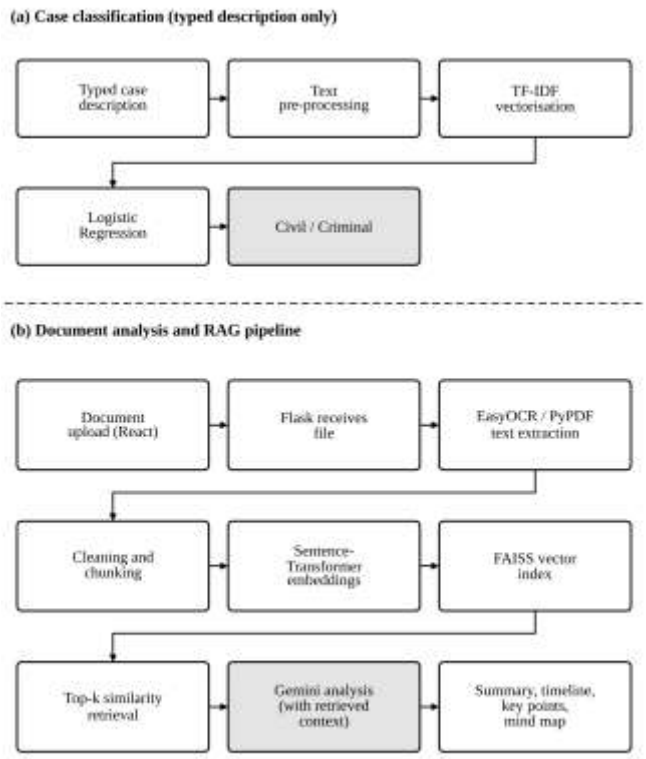


Fig. 2 (a) Case classification pipeline and (b) document analysis and RAG pipeline

D. Advocate Matching and Case Request

Using the predicted category and the advocate information stored in the database, the system lists suitable advocates. The client views the relevant advocate information and submits a case request. On the advocate dashboard, the advocate views received requests and accepts or rejects them; the backend validates each decision and updates the case status in the database After classification, the predicted case category is used along with advocate information stored in the database to display suitable advocates to the client. The client can view the available advocate details and select an appropriate advocate by submitting a case request.

The submitted request is displayed on the advocate dashboard, where the advocate can either accept or reject it. The Flask backend processes the selected action and updates the corresponding case status in the database. When a request is accepted, the case becomes available for further management by the advocate. This process establishes a structured connection between clients and advocates.. Accepted cases become available for case management.

E. Document Upload, OCR and Gemini Analysis

Clients can upload their legal documents through the frontend for preliminary examination. The Flask backend handles the document-upload and analysis requests through the respective routes.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

After receiving a document, the backend extracts its text using EasyOCR for scanned documents and PyPDF for PDF files containing extractable text. The extracted content is then provided to the Google Gemini API for analysis. The resulting extracted text and analysis are returned to the frontend for display.

Google Gemini generates four types of outputs:
1.**Document Summary:** Presents the main contents of the legal document in simpler language.
2.**Chronological Timeline:** Organises important events in the order in which they occurred.
3.**Key Points:** Identifies important information and details from the document.
4.**Mind Map:** Represents major topics and their relationships in a structured visual format.

The analysis results can also be compiled into a downloadable PDF report for reference by the advocate.

F. Retrieval-Augmented Generation

The Retrieval-Augmented Generation (RAG) module provides relevant legal information as additional context for Gemini during document analysis (Fig. 2b). Initially, PyPDF is used to extract text from the available legal reference PDF files. The extracted content is cleaned and divided into smaller text segments called chunks.

Sentence Transformers convert these chunks into numerical vector representations, known as embeddings. These vectors are then stored in a FAISS index to support efficient similarity-based searching. The legal reference collection currently indexed in the system includes selected sections from the Indian Penal Code (IPC), Code of Criminal Procedure (CrPC) and Bharatiya Nagarik Suraksha Sanhita (BNSS).

During analysis, the relevant input text is converted into an embedding using the same approach. FAISS compares this embedding with the stored vectors and retrieves the top-k most similar text chunks based on their similarity scores. The retrieved information is supplied to Google Gemini along with the document content as supporting context. This allows the generated analysis to use relevant information from the indexed legal references rather than depending only on the model's pre-trained knowledge [4].

G. Database and Case Status Tracking

Supabase PostgreSQL is used to maintain the application's data. The system primarily uses four tables: users, advocates, cases and case_documents. These tables maintain user details, advocate information, case records and information related to uploaded documents, respectively.

Once an advocate accepts a case request, the advocate can update its progress through different stages, such as Accepted, Documents Under Review, Case Analysis and Closed. Each status change is stored in the database and reflected on the client's dashboard. This enables clients to monitor the progress of their cases without repeatedly contacting the advocate for updates.

H. End-to-End Workflow

Figure 3 illustrates the overall working process of KanoonSamparka. Initially, the client registers and logs in to the application before entering a description of the legal problem. The system classifies the description into either the Civil or Criminal category and displays suitable advocates based on the predicted category and available advocate information.

The client then submits a case request to the selected advocate. The advocate reviews the request and either accepts or rejects it. Once accepted, the client can upload the relevant legal documents for processing. The uploaded documents pass through the OCR, RAG and Gemini-based analysis pipeline to generate a summary, timeline, key points and mind map. The advocate can then manage the accepted case and update its progress, while the client can view the latest case status through the dashboard.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

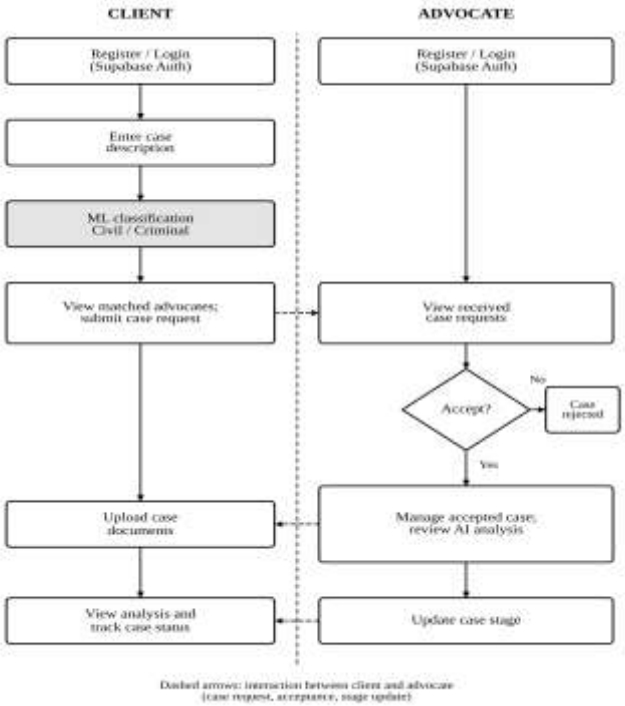


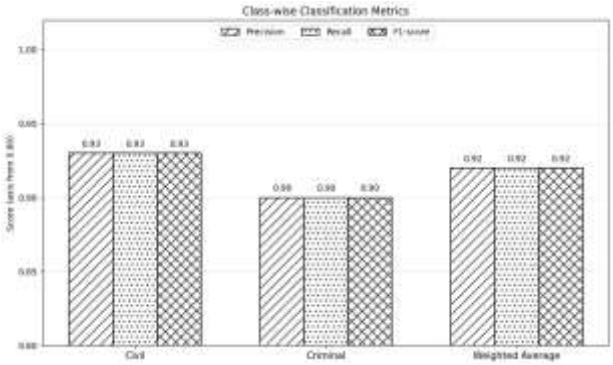
Fig. 3 End-to-end client and advocate workflow

V. RESULTS AND DISCUSSION

A. Classification Performance

The Civil/Criminal classification model was developed to classify typed case descriptions into two categories: Civil and Criminal. The model uses TF-IDF for text feature extraction and Logistic Regression for prediction. Its performance can be evaluated using precision, recall and F1-score for both categories, along with a confusion matrix, as represented in Fig. 4 and Table II. These evaluation measures help determine how effectively the model distinguishes between the two case types.

The classification process uses only the case description entered by the client. Uploaded documents, OCR-extracted text and Gemini-generated analysis are not used as inputs to the classification model. This separation ensures that document processing and AI-based analysis do not influence the Civil/Criminal prediction.



IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

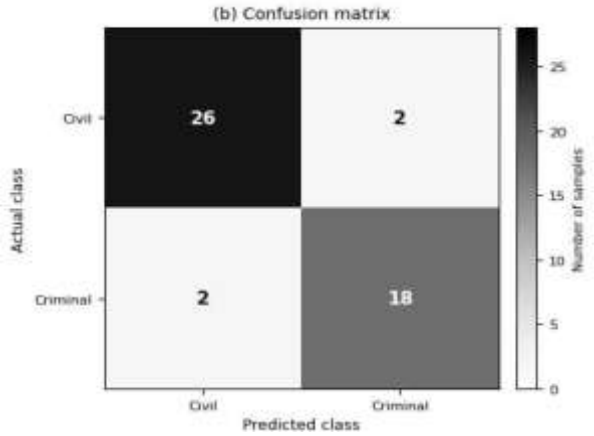


Fig. 4 (a) Class-wise precision, recall and F1-score; (b) confusion matrix

TABLE II CLASSIFICATION REPORT ON THE TEST SET

Class	Precision	Recall	F1-score	Support
Civil	0.93	0.93	0.93	28
Criminal	0.90	0.90	0.90	20

B. Document Processing Time

Fig. 5(a) presents the average time required by each stage of the document-processing pipeline. The processing stages include OCR/PDF text extraction, text cleaning, chunking, embedding generation, FAISS retrieval and Gemini analysis. The overall processing time depends on the document size and the operations performed during analysis.

Fig. 5(b) illustrates the relationship between document length and processing time. Scanned documents may require additional processing because OCR must identify and extract text from document images. Similarly, lengthy documents may take more time to process and analyse. The time required for Gemini analysis may also vary depending on the amount of extracted text and the response time of the external service. These stages help identify the factors that influence the overall document-processing time.

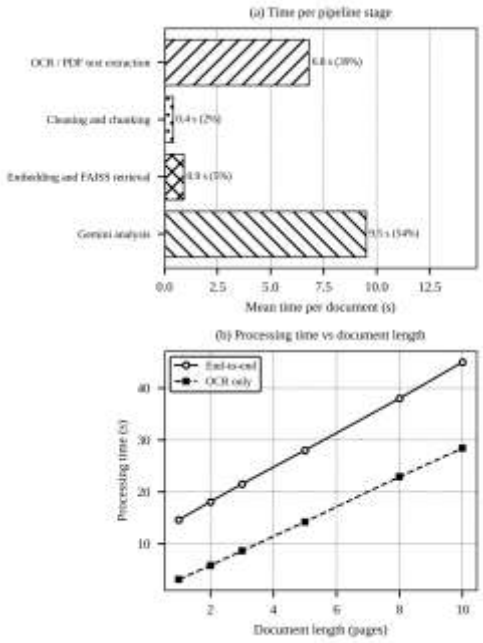


Fig. 5 (a) Mean time per pipeline stage; (b) processing time versus document length

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511216

C. Retrieval Quality and Backend Response Time

The Retrieval-Augmented Generation (RAG) module uses Sentence Transformers to convert text into embeddings and FAISS to retrieve relevant legal information through similarity search. The retrieval process can be evaluated by checking whether relevant legal sections appear among the top-k results returned by FAISS, as illustrated in Fig. 6(a). Retrieving multiple relevant text chunks can provide Gemini with additional context for generating its responses.

The relevance of the retrieved information depends on the quality of the legal reference documents, text extraction and similarity matching. The retrieved text chunks are supplied to Google Gemini as supporting context during document analysis. However, the relevance and correctness of the generated output must still be checked.

Fig. 6(b) presents the backend response time for different application operations, such as user login and case classification. The backend also handles document uploading, case management and communication with the database. Response times may vary depending on server performance, database communication, document size and external AI service response times. These operations contribute to the overall responsiveness of the application.

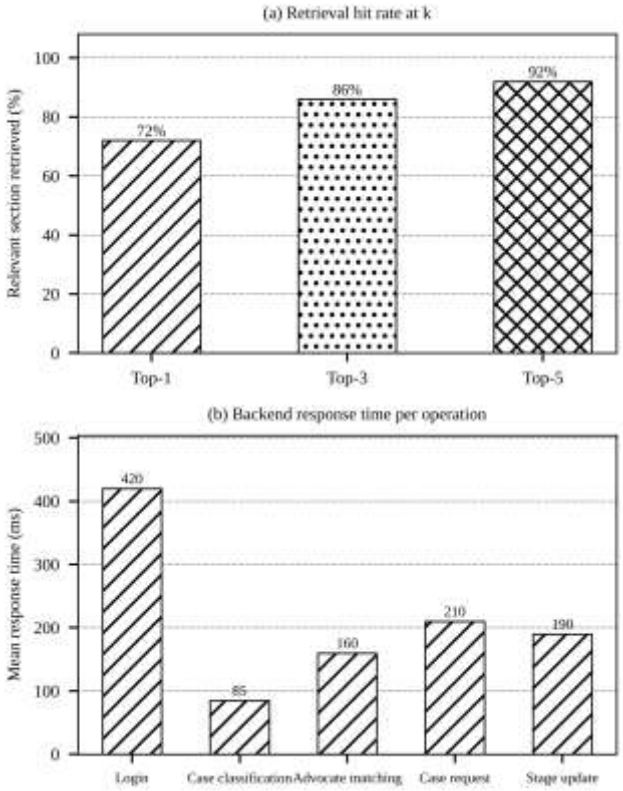


Fig. 6 (a) Retrieval hit rate at k = 1, 3 and 5; (b) mean backend response time per operation

D. Discussion and Limitations

The proposed system combines TF-IDF and Logistic Regression to classify typed case descriptions into the two broad categories, Civil and Criminal. Keeping the classification module separate from document analysis allows the model to perform predictions without depending on OCR processing or Gemini-generated outputs. The evaluation measures represented in Fig. 4 and Table II can be used to assess the classification performance of the model.

The document-processing module helps organise information from uploaded legal documents through text extraction and AI-assisted analysis. As illustrated in Fig. 5, the processing stages include text extraction, cleaning, chunking, embedding generation, retrieval and Gemini analysis. In addition, the RAG approach retrieves relevant legal reference information to provide supporting context for Gemini-generated responses. Fig. 6 represents the retrieval process and backend response-time evaluation.

The system has certain limitations. First, the classification model is restricted to two broad legal categories and may not identify more specific legal issues. Second, the accuracy of OCR-based text extraction depends on the quality, clarity and readability of uploaded documents. Third, processing time may increase for lengthy documents, particularly when

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511216

scanned pages require OCR. Finally, the summaries, timelines, key points and mind maps generated by Google Gemini may not always contain accurate information. Therefore, these outputs should be reviewed and verified by the advocate before being used for further case-related work.

KanoonSamparka is intended to support preliminary legal document examination and improve communication between clients and advocates. It does not replace professional legal judgement, provide legal advice or predict case outcomes.

VI. CONCLUSION

This paper presented KanoonSamparka, a web-based platform designed to simplify the initial stages of legal case management by providing a common interface for clients and advocates. The system combines machine learning-based case classification, Optical Character Recognition (OCR), Retrieval-Augmented Generation (RAG), and Gemini-based document analysis to support preliminary case understanding and information management. Separate dashboards for clients and advocates help organize their respective activities, while the structured case request, acceptance, and status-update process allows clients to follow the progress of their cases more transparently. The platform also provides downloadable reports containing relevant case information and document analysis results.

The developed system demonstrates how machine learning and AI-based technologies can be integrated with document processing and case management features to make legal information more accessible and organized. However, KanoonSamparka is intended to assist with preliminary case understanding and workflow management rather than replace professional legal advice or the judgement of practising advocates. Future enhancements may include testing the system with larger and more diverse real-world case datasets, improving OCR performance when processing low-quality or unclear scanned documents, expanding the classification model to support additional case categories, and conducting usability evaluations with practising advocates and clients. These improvements could help enhance the system's reliability, usability, and effectiveness in supporting digital legal case management.

REFERENCES

[1]. H. Zhong, C. Xiao, C. Tu, T. Zhang, Z. Liu, and M. Sun, "How does NLP benefit legal system: A summary of legal artificial intelligence," in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics (ACL), Online, Jul. 2020, pp. 5218–5230.
[2]. V. Malik et al., "ILDC for CJPE: Indian legal documents corpus for court judgment prediction and explanation," in Proc. 59th Annu. Meeting ACL and 11th Int. Joint Conf. Natural Language Processing (Vol. 1: Long Papers), Online, Aug. 2021, pp. 4046–4062.
[3]. I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The Muppets straight out of law school," in Findings of the Assoc. Comput. Linguistics: EMNLP 2020, Online, Nov. 2020, pp. 2898–2904.
[4]. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459–9474.
[5]. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, Hong Kong, Nov. 2019, pp. 3982–3992.
[6]. J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, Jul. 2021.
[7]. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[8]. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manag., vol. 24, no. 5, pp. 513–523, 1988.
[9]. Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv:2312.11805, 2023.
[10]. JaidedAI, "EasyOCR: Ready-to-use OCR with 80+supported languages," GitHub repository. [Online]. Available: https://github.com/JaidedAI/EasyOCR

 This work is licensed under a Creative Commons Attribution 4.0 International License