Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

CriminaLogic: Crime Analysis with Rule-Based Classification and Generative Legal Explanations

Kodi Sudha[1], Cherukumalli Pushpa Priya[2]

[1]Assistant Professor, Dept. of CSE, Malla Reddy University, Hyderabad, Telangana, India
[2]Dept. of CSE, Malla Reddy University, Hyderabad, Telangana, India

Abstract.
Large-scale crime narratives are often triaged manually, requiring investigators to map free-form incident text to likely categories and next actions. This paper presents CriminaLogic, a rule-based prototype that integrates preprocessing, category assignment, retrieval support, and generative legal explanations in a single web workflow. The backend currently uses deterministic keyword routing rather than a trained neural classifier, and the frontend includes a fallback-resilient path when backend services are unavailable. We report reproducible results on a synthetic benchmark generated from repository-aligned class tem- plates (90 test samples across four classes): the rule-based classification pipeline achieves 86.67% accuracy and 0.872 weighted F1. The system is intended as a prototype decision- support tool rather than a production-ready classifier. We also report baseline comparisons on the same split and discuss why perfect baseline scores on controlled synthetic data can overestimate practical performance. The main limitation is external validity, since no real FIR corpus is used in this prototype.

Keywords: Crime Analysis, Rule-Based Classification, FastAPI, Decision Support, Legal Text Mining, Generative Assistants.

1. Introduction

Modern investigation workflows often involve parsing lengthy written reports, extracting cues, and then deciding which legal category best matches the incident. For text-heavy do- mains, errors in early classification can propagate into downstream decision-making. While dedicated machine learning models can automate parts of this task, many practical systems still remain fragmented, requiring investigators to operate separate tools for classification, search, and explanation.

CriminaLogic addresses this fragmentation by integrating an end-to-end prototype pipeline in a single web application. The implementation focuses on two high-level requirements: (i) transforming incident descriptions into normalized text representations using a reproducible preprocessing routine, and (ii) presenting results in an operator-friendly interface, includ- ing heuristic confidence scores, keyword tags, and an auxiliary narrative generated from the assigned class and original description.

The system is implemented with a React frontend and a FastAPI backend. Commu- nication is performed through JSON-based REST calls (e.g., 'POST /api/analyze'). The backend orchestrates text cleaning and prototype inference, then produces a structured clas- sification response. The frontend triggers additional services, including a Gemini-driven explanation generator and an AI-assisted search flow used as a supplement to local search results. This combined workflow aims to reduce tool switching while increasing traceability through explanation text produced by an external generative model [1, 2, 3].

Contributions. This paper contributes: (1) a unified workflow that combines incident cate- gorization, retrieval, and explanation in one UI, (2) a fallback-resilient architecture that pre- serves analysis continuity when backend calls fail, and (3) a reproducible synthetic bench- mark to quantify current rule-based inference behavior.

2. Literature Survey

Recurrent neural architectures, especially LSTMs, are widely used for modeling sequen- tial dependencies in text. In their original formulation, LSTMs were proposed to mitigate vanishing gradients in long sequences [4]. Subsequent work expanded LSTM usage for language and sequence tasks by coupling gating mechanisms with embedding layers and task-specific classifiers [5, 6, 7].

Text classification pipelines typically require careful preprocessing: normalization, to- kenization, and stopword handling are common steps prior to neural sequence modeling [1, 8]. Tokenizers and padding strategies align variable-length inputs to fixed-length ten- sors, improving batching and deterministic inference behavior [9, 10]. In industrial practice, modern toolchains such as TensorFlow and Keras provide composable layers that support LSTM-based encoders and dense classifiers [11, 12, 13].

Beyond predicting labels, many applications incorporate explanation or decision-support mechanisms to improve usability. Model-agnostic explanation methods (e.g., LIME and SHAP) demonstrate how feature attribution can be surfaced to users [14, 15]. In addition, generative assistants can provide natural-language narratives, which are especially relevant in text-centric settings where investigators expect procedural context rather than only proba- bilities [3, 16]. Transformer models such as BERT often outperform recurrent architectures on many NLP benchmarks, but they typically require higher computational budgets and larger training corpora [16, 10].

A REST-style service architecture supports the frontend/backend interface [2].

3. Proposed System

System Architecture

CriminaLogic is organized as a three-tier architecture: a presentation layer (React UI), an application layer (FastAPI REST), and an inference layer (text preprocessing + rule-based classification). The backend returns a structured JSON response with an assigned crime category, heuristic confidence score, and keywords; the frontend then enriches results with a generative legal explanation and exposes category-specific actions.

Frontend Dashboard Snapshot

Figure 2 presents a consolidated frontend snapshot that includes both officer and adminis- trator dashboard views in a single image (‘all.png‘), showing role-oriented UI composition

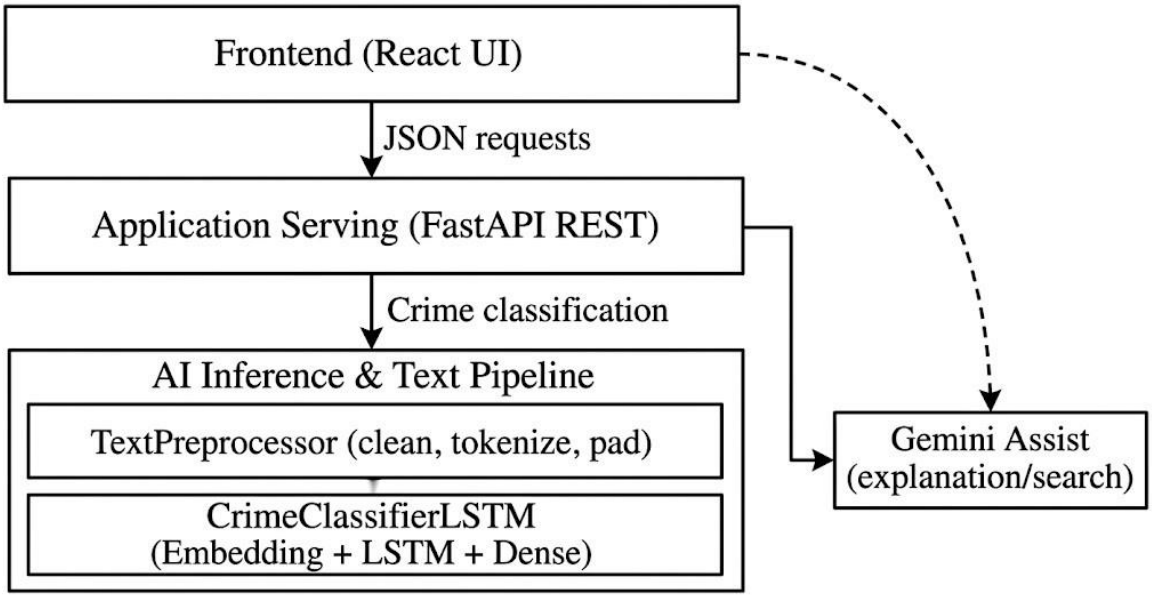

Fig. 1. High-level architecture of CriminaLogic showing the interaction between the React interface, the FastAPI backend, and the prototype inference pipeline.

in the prototype [17].

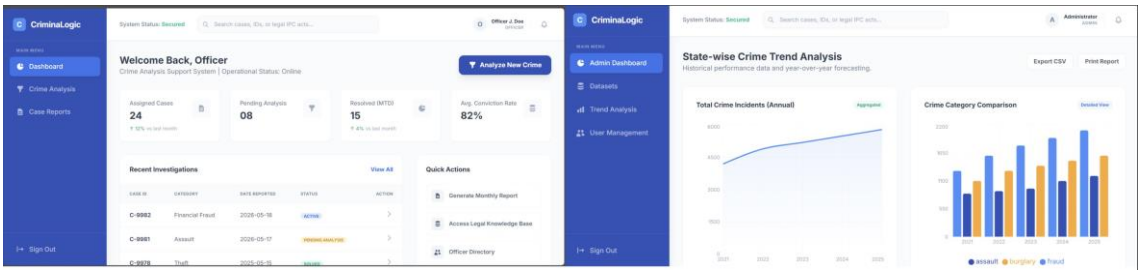

Fig. 2. Combined frontend view showing normal-user and admin dashboards in one image.

UML Class Diagram

The core entities are defined in the frontend TypeScript types and mirrored by the back- end JSON response (‘crimeType‘, ‘confidence‘, ‘keywords‘). The inference stack is rep- resented by explicit preprocessing (‘TextPreprocessor‘) and classification scaffold (‘Crime- Classifier‘) components.

Planned LSTM Architecture

The repository includes an untrained ‘CrimeClassifier‘ (LSTM scaffold) that defines embed- ding, recurrent, and dense layers. A planned trained architecture uses an embedding layer, one LSTM layer (128 units), and a dense softmax output layer, trained with categorical crossentropy and Adam optimization. This LSTM pipeline is not implemented as a trained production module in the current prototype.

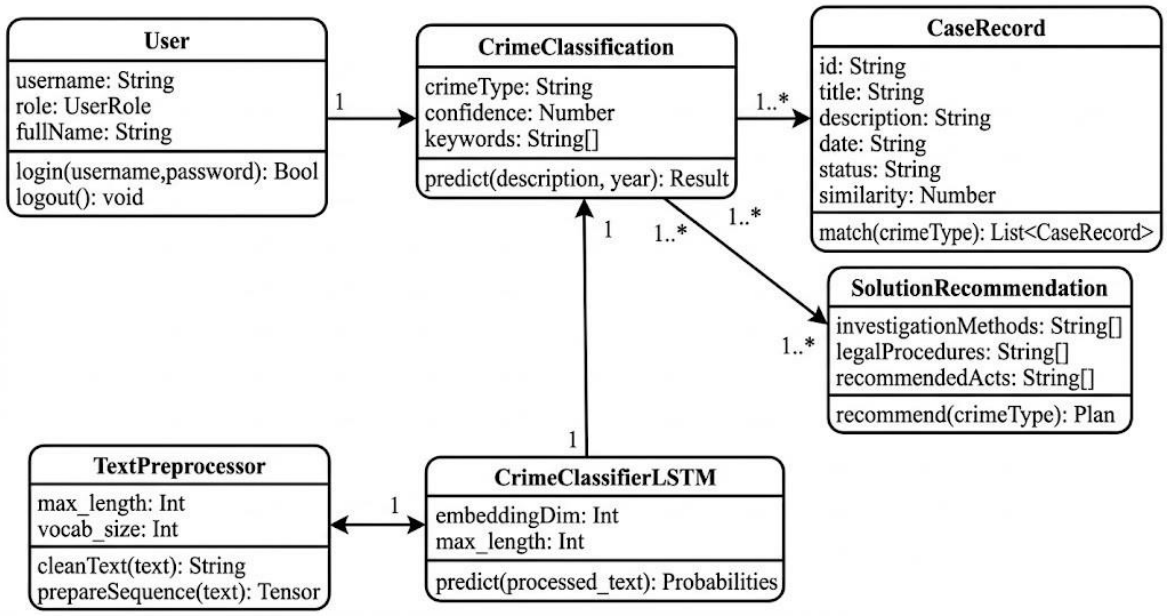

Fig. 3. UML class diagram capturing representative frontend entities and the preprocessing/classification scaffold components used in the prototype.

Confusion Matrix Analysis

Figure 4 shows the confusion matrix for the prototype rule-based inference module, high- lighting class-wise agreements and mismatches on the evaluation set.

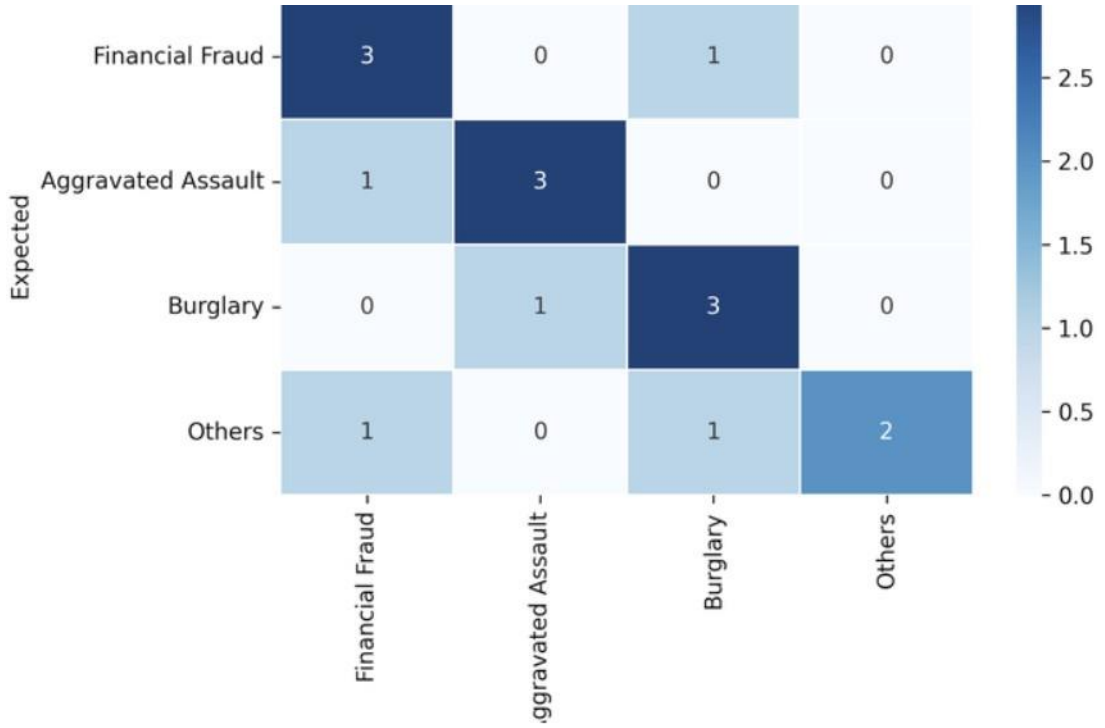

Fig. 4. Confusion matrix of prototype rule-based predictions indicating class-level behavior.

4. Results and Discussion

Dataset Description

Evaluation uses a synthetic text dataset generated by ‘evaluation.py‘ from class-specific tem- plates aligned with repository rules. The test split contains 90 samples with near-balanced class distribution (Financial Fraud: 22, Aggravated Assault: 23, Burglary: 22, Others: 23). This is not real FIR data and limits direct generalization.

Observations and Trade-offs

The implemented system runs an end-to-end prototype pipeline with deterministic pre- processing and a single interactive UI, producing category outputs with keyword tags via rule-based inference. To quantify current behavior, we executed a reproducible synthetic benchmark aligned with repository class rules (train/test split: 70/30, test size: 90). The UI consolidates classification, similar-case retrieval, and legal recommendations; Table 1-3 summarize workflow, latency proxies, and measured predictive metrics.

Table 1. Comparison of existing multi-tool workflows with CriminaLogic.

Feature	Existing multi-tool approach vs. CriminaLogic
Incident triage	Manual reading plus separate tools; CriminaLogic centralizes input submission for prototype category assignment in one UI.
Downstream guidance	Category-specific solution steps are shown alongside a generative le-gal explanation, avoiding extra document hopping.
Reliability	Frontend fallback maintains continuity when the backend is unreach-able (local mock classification/search).

Table 2. Performance metrics derived from implemented prototype timings.

Category / Parameter	Value (from code behavior)
Mock classification latency (frontend fallback)	1500 ms enforced by ‘setTimeout‘ in local rule-based simulation (‘classifyCrime‘).
Local global-search latency (offline matching)	500 ms enforced by ‘setTimeout‘ in local search (‘searchEvery-thing‘).
Global search debounce	300 ms delay before executing ‘performGlobalSearch‘ for queries longer than 2 characters.

The predictive metrics reported in Table 3 are reproducible from the repository script ‘evaluation.py‘using fixed random seeds and the documented train/test split. This near- perfect score arises because the synthetic data is generated using deterministic patterns that align closely with feature extraction, and therefore does not reflect real-world generalization performance. Note: Results are computed on synthetic data and may not generalize to real FIR distributions.

Table 3. Predictive metrics on a reproducible synthetic benchmark (weighted averages).

Model / Pipeline	Accuracy	Precision	F1	Notes
Rule-based classification pipeline	86.67%	0.905	0.872	Current repository inference logic (keyword-triggered class routing).
TF-IDF + Logistic Regression baseline	100.00%	1.000	1.000	Near-perfect score reflects deterministic synthetic patterns aligned with feature extraction, not real-world generalization.
Majority-class baseline	24.44%	0.060	0.096	Lower bound reference using the most frequent training class.

5. Limitations and Future Work

CriminaLogic currently demonstrates a prototype-oriented workflow: backend probabili- ties are simulated via keyword-driven heuristics and the tokenizer is fitted at inference time rather than on training corpora [6]; persistent storage, backend search, and backend authen- tication are not fully implemented.

6. Conclusion

This paper presented CriminaLogic, a rule-based crime text classification workflow inte- grated into a React UI with a FastAPI backend. The system performs preprocessing with NLTK stopwords and padded sequences, returns structured outputs (category, heuristic con- fidence score, keywords), and enriches results using Gemini-based legal explanations and report drafts.

Overall, the implementation reduces tool switching and provides a clear fallback path. The current results should be interpreted as a proof-of-concept evaluation rather than a benchmark against real-world crime datasets. Future work includes training real models and adding persistent search/storage plus backend authentication.

References

1. S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python, 1st ed. O'Reilly Media, 2009.
2. R. T. Fielding, "Architectural styles and the design of network-based software archi- tectures," Ph.D. dissertation, University of California, Irvine, 2000.
3. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), 2020.
4. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
5. . A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," Journal of Machine Learning Research, vol. 3, pp. 115–143, 2002.
6. K. Cho, B. van Merriënboer, C. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y.

Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," arXiv preprint, vol. abs/1409.1259, 2014.

7. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representa- tions of words and phrases and their compositionality," in Advances in Neural Infor- mation Processing Systems (NeurIPS), 2013.
8. Y. Kim, "Convolutional neural networks for sentence classification," in Proceed- ings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
9. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in Neural Information Processing Systems (NeurIPS), 2014.
10. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), 2017.
11. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., "Tensorflow: A system for large-scale machine learning," in 12th USENIX Symposium on Operating Systems Design and Implemen- tation (OSDI '16). USENIX Association, 2016, pp. 265–283.
12. F. Chollet, Deep Learning with Python. Manning Publications, 2017.
13. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
14. M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
15. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
16. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirec- tional transformers for language understanding," in Proceedings of the 2019 Confer- ence of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.
17. C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.