Combining Extractive and Generative Methods for Legal Summarization: Tayronas Trigrams at JUST-NLP 2025

Erich Giusseppe Soto Parada
Universidad de los Andes
Bogotá, Colombia
eg.soto@uniandes.edu.co

David Cuevas Alba
Universidad de los Andes
Bogotá, Colombia
d.cuevas@uniandes.edu.co

Carlos Manuel Muñoz Almeida
Universidad de los Andes
Bogotá, Colombia
c.munoza@uniandes.edu.co

Abstract

This paper presents Tayronas Trigrams's methodology and findings from our participation in the JUST-NLP 2025 Shared Task of Legal Summarization (L-SUMM), which focused on generating abstractive summaries of lengthy Indian court judgments. Our initial approach involved evaluating and fine-tuning specialized sequence-to-sequence models like Legal-Pegasus, Indian Legal LED, and BART. We found that these small generative models, even after fine-tuning on the limited InLSum dataset (1,200 training examples), delivered performance (e.g., Legal-Pegasus AVG score: 16.50) significantly below expected.

Consequently, our final, best-performing method was a hybrid extractive-abstractive pipeline. This approach first employed the extractive method PACSUM to select the most important sentences yielding an initial AVG score of 20.04 and then utilized a Large Language Model (specifically Gemini 2.5 Pro), correctly prompted, to perform the final abstractive step by seamlessly stitching and ensuring coherence between these extracted chunks. This hybrid strategy achieved an average ROUGE-2 of 21.05, ROUGE-L of 24.35, and BLEU of 15.12, securing 7th place in the competition. Our key finding is that, under data scarcity, a two-stage hybrid approach dramatically outperforms end-to-end abstractive fine-tuning on smaller models.

1 Introduction

The legal systems of highly populous nations, such as India, are facing a critical challenge due to judicial pendency. As reported by the National Judicial Data Grid (NJDG), India alone contends with over 44 million pending cases across its courts. This massive backlog, often caused by manual, inefficient document processing, delays timely justice and undermines the fundamental rights the system is designed to protect. Automated systems powered by Natural Language Processing (NLP) offer a scalable solution to assist legal professionals, streamline document workflow, and ultimately improve public access to case information.

Our team, Tayronas Trigrams, participated in the JUST-NLP 2025 Shared Task 1: Legal Summarization (L-SUMM), which required generating 500-word abstractive summaries for lengthy Indian court judgments. We found that the task's inherent challenges, such as nuanced legal reasoning and textual abstraction, were significantly amplified by the limited InLSum dataset (1,200 training instances). This data scarcity caused existing specialized small generative models to struggle, motivating our pivot from a purely abstractive approach to a novel, two-stage methodology.

- We demonstrate the limited efficacy of fine-tuning small, specialized transformer models (e.g., Legal-Pegasus, Indian Legal LED, BART) for abstractive legal summarization under low-resource conditions.
- We establish that a simple extractive baseline significantly outperforms these fine-tuned generative models when data is scarce.
- We propose and validate a Hybrid Extractive-Abstractive pipeline, which uses Large Language Models (LLMs) to connect and coherently refine extracted content, resulting in a substantial performance gain.
- Our final model achieved strong competitive metrics (ROUGE-2: 21.05, ROUGE-L: 24.35) by leveraging this hybrid approach, underscoring its superiority for low-data, domain-specific summarization.

2 Related Work

Automatic legal summarization is challenging due to extreme document length (Shukla et al., 2022), domain-specific "legalese" (Joshi et al., 2024), and the need for high factual consistency. Indian legal texts present additional challenges, being "noisier and poorly organized" (Sharma et al., 2023). While traditional extractive methods like LexRank are factually robust (Shukla et al., 2022; Sharma et al., 2023), recent work has shifted to abstractive models like BART (Shukla et al., 2022) and specialized models such as Legal-Pegasus. Notably, Sharma et al. (2023) identified Legal-Pegasus as a top performer on Indian legal data, justifying its use as a strong baseline.

A primary obstacle, however, is data scarcity. The L-SUMM task utilizes the low-resource InL-Sum dataset, creating a significant performance bottleneck. This setting favors robust baselines; Shukla et al. (2022) observed that strong extractive models can perform on par with abstractive ones. Furthermore, Joshi et al. (2024) found that even large LLMs (e.g., GPT-4) can underperform fine-tuned models on this specific task (SUMM). Our work builds directly on these findings, corroborating the limitations of fine-tuning small models in a low-resource setting and instead proposing a hybrid approach that combines extractive content selection with LLM-based refinement.

3 Methodology

Our methodology was structured as a multi-stage process, beginning with a systematic evaluation of existing models and culminating in a hybrid approach. The initial phase, detailed below, focused on benchmarking specialized pre-trained models to establish a performance baseline on the L-SUMM task, following similar comparative analyses in the literature (Sharma et al., 2023; Shukla et al., 2022). This foundational analysis was critical in guiding our subsequent, more complex strategies.

3.1 Initial Model Evaluation

To assess the zero-shot capabilities of existing models on the provided dataset, we implemented a unified evaluation pipeline using the Hugging Face Transformers library. This allowed us to systematically compare promising pre-trained, domain-specific models for legal text summarization (Sharma et al., 2023; Shukla et al., 2022). The selected models were:

- **BART** (`sanatann/legal-summarizer-bart`): A model based on the BART architecture, fine-tuned for legal summarization, commonly used as a baseline for this task (Sharma et al., 2023; Shukla et al., 2022).
- **LED** (`TheGod-2003/legal-summarizer`): A Longformer-Encoder-Decoder (LED) model designed to handle long documents, making it suitable for legal texts (Sharma et al., 2023; Shukla et al., 2022).
- **Indian Legal LED** (`Yashaswat/indian-legal-led-base`): An LED model specifically fine-tuned on Indian legal documents. The use of a specialized Legal-LED is supported by Joshi et al. (2024) and Shukla et al. (2022), who identify it as a strong-performing model for the domain.
- **Legal Pegasus** (`nsi319/legal-pegasus`): A model based on the Pegasus architecture adapted for the legal domain. This model was prioritized for evaluation as Sharma et al. (2023) found it to “outperform over all other models” for Indian legal summarization, a finding supported by Shukla et al. (2022).

Our pipeline loaded each model and its corresponding tokenizer, processed documents from the training set, generated summaries, and evaluated them against the ground truth. Performance was measured using standard summarization metrics: ROUGE-2, ROUGE-L, and BLEU, which are standard for the task and the In-Abs dataset (Joshi et al., 2024; Shukla et al., 2022). This initial evaluation provided the baseline data that informed our decision to move away from a purely fine-tuning-based approach.

3.1.1 Initial Evaluation Results

The initial evaluation was conducted on a small, representative sample of 8 documents from the training set to quickly gauge the zero-shot performance of each model. While not statistically significant, this preliminary analysis provided valuable directional insights into which models were most promising. The results, shown in Table 1, indicated that Legal Pegasus offered a competitive starting point.

3.2 Fine-Tuning Phase

3.2.1 Motivation and Model Selection

Following Bhattacharya et al. (2021), who recommended Legal-Pegasus for legal summarization via

Model	ROUGE-2	ROUGE-L	BLEU
BART	9.34	17.37	10.06
LED	11.77	18.91	16.30
Indian Legal LED	2.45	16.67	7.19
Legal Pegasus	20.05	17.37	10.06

Table 1: Preliminary zero-shot results on a sample of 8 documents. Scores are corpus-level.

chunking, we selected `nsi319/legal-pegasus` as our base model. However, Indian legal judgments averaged 10,000+ tokens, exceeding Legal-Pegasus's 1,024-token limit, necessitating an intelligent chunking strategy. We investigated whether embedding-based similarity could improve chunk-summary alignments.

3.2.2 Chunking Strategy

We adopted the chunking approach of Bhattacharya et al. (2021), inspired by Gidiotis and Tsoumakas (2020), which segments documents into chunks and constructs targeted summaries by mapping summary sentences to similar document sentences. Given a document-summary pair (d, s), we partition d into chunks $\{d_1, d_2, \ldots, d_n\}$ and generate chunk-specific summaries s_i by aggregating summary sentences that map to sentences within each chunk d_i (see Appendix A for detailed methodology). We compared two similarity metrics while keeping the fine-tuning pipeline constant.

Experiment 1: TF-IDF Baseline. Our baseline employed TF-IDF cosine similarity with fixed-size chunks (400 words, 50 overlap) and a 0.1 similarity threshold, yielding 4,706 training pairs. While computationally efficient, this bag-of-words approach lacks semantic context (detailed vectorization parameters in Appendix A).

Experiment 2: MCS-SBERT Optimization. Building on Mean Cosine Similarity (MCS) (Reimers and Gurevych, 2019), we replaced TF-IDF with dense embeddings from `all-MiniLM-L6-v2` Sentence-BERT (384-dim), which capture semantic relationships beyond lexical overlap. We implemented semantic chunking respecting paragraph boundaries (500 words, 40-word minimum) and raised the similarity threshold to 0.4 to filter false positives. Quality filters (compression ratio 0.05–0.4, minimum summary length 40 words) excluded degenerate pairs, yielding 638 high-confidence examples—a deliberate trade-off sacrificing quantity for quality.

Analysis. Table 2 shows MCS-SBERT outperformed TF-IDF with 82% fewer pairs, attributable to semantic capture (SBERT encodes meaning beyond surface similarity) and noise reduction (elevated threshold filtered spurious alignments). This validates that in low-resource legal domains, curated high-precision data outweighs large volumes of noisy examples.

3.2.3 Fine-tuning Setup

We fine-tuned Legal-Pegasus using Hugging Face Transformers with the 638 MCS-SBERT pairs (85/15 split). Configuration: LR 2×10^{-5}, 500 warmup steps, batch 4 (effective: 8), 3 epochs, 1,024/512 tokens, BF16 on A100 GPU, beam search (6). Training converged in 45 minutes with ROUGE evaluation (Lin, 2004).

3.2.4 Results and Analysis

Our fine-tuned Legal-Pegasus achieved ROUGE-2: 17.1 and ROUGE-L: 19.8, validating MCS-SBERT's effectiveness but falling below our extractive baseline (ROUGE-2: 20.51, ROUGE-L: 23.49, as shown in Table 3). This gap reveals limitations of fine-tuning small models in low-resource legal scenarios: (1) *data scarcity*—638 pairs vs. 10,000+ typically required (Zhang et al., 2020); (2) *domain complexity*—legal precision demands exceed small model capacity (568M parameters), causing occasional hallucinations; and (3) *abstractive risk*—semantic drifts unacceptable for legal fidelity, where extractive methods excel.

Despite suboptimal performance, this validated dense embeddings' superiority over bag-of-words and demonstrated fine-tuning's viability threshold, motivating our pivot to a hybrid extractive-abstractive approach.

3.3 Hybrid Extractive-Abstractive Method

We propose a two-stage approach combining extractive pre-selection with abstractive refinement for legal document summarization. Legal case judgments pose significant challenges due to extreme length (10,000+ tokens), technical terminology, and complex argumentation structures (Bhattacharya et al., 2021).

3.3.1 Final method

Stage 1: Extractive Pre-selection. Documents are first segmented into semantic chunks (maximum 512 tokens per chunk, no overlap), breaking at natural boundaries to preserve meaning. We apply PACSUM (Zheng and Lapata, 2019), a BERT-

Table 2: Comparative study of chunking strategies. MCS-SBERT achieves superior performance with 82% fewer training pairs. The performance is reported on the InLSum Validation Set (200 datapoints).

Method	Similarity	Threshold	Pairs	AVG	R-2	R-L	BLEU
TF-IDF Baseline	Cosine (BoW)	0.1	4,706	16.77	19.94	22.51	7.85
MCS-SBERT	SBERT Cosine	0.4	638	**17.16**	**20.64**	**21.62**	**9.23**

based unsupervised method that constructs directed sentence graphs with position-augmented centrality scoring. PACSUM ranks sentences within each chunk by semantic similarity and positional importance. We then select the chunks corresponding to the highest PACSUM scores and aggregate them sequentially until reaching a 1000-token budget, adding additional highly-ranked content when space permits. This reduces input length by 60-70% while retaining salient information.

Stage 2: Abstractive Refinement. The aggregated 1000-token extractive summary is processed by Gemini 2.5 Pro (DeepMind, 2024) using zero-shot prompting with optimized instructions. We employ automated prompt optimization through iterative refinement, evaluating candidate prompts on validation examples and selecting those maximizing ROUGE scores. While we experimented with few-shot learning (1-3 demonstration examples), zero-shot prompting consistently outperformed few-shot across all metrics. The LLM generates coherent abstractive summaries by paraphrasing, fusing sentences, and removing redundancy.

3.3.2 Experimental Results

We evaluate our method on the Indian Court Judgment Summarization shared task. Table 3 presents performance across four metrics: the AVG Score (primary ranking metric), ROUGE-2 ($R2$), ROUGE-L (RL), and BLEU (B). The AVG Score (C) is calculated as the average of the three standard relevance metrics:

$$\text{AVG Score } (C) = \frac{R2 + RL + B}{3}$$

Method	Score	R-2	R-L	BLEU
PACSUM (extractive)	19.61	20.51	23.49	14.84
PACSUM + Gemini 2.5 Pro	**20.17**	**21.05**	**24.35**	**15.12**

Table 3: Performance comparison on test set. PACSUM processes 512-token chunks and aggregates to 1000 tokens. The hybrid method shows consistent improvements across all metrics.

The hybrid approach outperforms the purely extractive baseline across all metrics, with improvements of +0.56 points in Competition Score, +0.54 in ROUGE-2, +0.86 in ROUGE-L, and +0.28 in BLEU. These gains demonstrate that abstractive refinement by the LLM adds value beyond extractive selection alone.

Analysis. The modest improvements suggest that the extractive pre-selection already captures most salient content effectively. The LLM's primary contribution is enhancing fluency and coherence rather than identifying additional key information. The two-stage strategy (512-token chunks for processing, 1000-token aggregation for LLM input) proved critical for managing computational costs while maintaining summary quality. Notably, zero-shot prompting outperformed few-shot approaches, suggesting that well-crafted instructions are more effective than demonstration examples for this task.

4 Conclusion

In this paper, we described the methodology used by Tayronas Trigrams for the L-SUMM task at JUST-NLP 2025. Our initial experiments focused on fine-tuning specialized models like Legal-Pegasus, but we found that they did not perform well due to the small size of the InLSum dataset (1,200 examples). The models struggled to generate accurate summaries without more training data.

To solve this, we developed a hybrid pipeline. We used PACSUM for extractive selection to identify the most relevant sentences, followed by Gemini 2.5 Pro for abstractive refinement. Our approach achieved 7th place in the competition with a ROUGE-L score of 24.35. These results indicate that when data is limited, combining extractive methods with Large Language Models is a practical and effective strategy for legal summarization. Future work will explore dynamic chunking strategies to mitigate information bottlenecks and evaluate open-source models to address the privacy limitations of proprietary APIs.

Limitations

While our hybrid extractive-abstractive approach proved effective for the shared task, we acknowledge three primary limitations:

- **Dependency on Proprietary APIs:** Our reliance on Gemini 2.5 Pro introduces costs and potential reproducibility issues. Furthermore, sending legal documents to external APIs raises data privacy concerns that purely local models avoid.
- **Information Bottleneck:** The abstractive generator is strictly limited by the initial extractive step. If the PACSUM algorithm fails to select a crucial piece of evidence or legal precedent, the LLM has no way to recover that information, as it never sees the full original document.
- **Domain Specificity:** Our optimization steps, particularly the chunking thresholds and prompt engineering, were tailored specifically for Indian case law. These parameters may not generalize effectively to other legal systems or languages without significant adjustment.

References

Paheli Bhattacharya, Kaustubh Hiware, Subham Rajgaria, Nilay Pochhi, Kripabandhu Ghosh, and Saptarshi Ghosh. 2021. A comparative study of summarization algorithms applied to legal case judgments. In *Proceedings of the 43rd European Conference on Information Retrieval*, pages 413–428. Springer.

Google DeepMind. 2024. Gemini: A family of highly capable multimodal models. https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/. Accessed: 2024-12-15.

Alexios Gidiotis and Grigorios Tsoumakas. 2020. A divide-and-conquer approach to the summarization of long documents. In *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, volume 28, pages 3029–3040. IEEE.

Abhinav Joshi, Shounak Paul, Akshat Sharma, Pawan Goyal, Saptarshi Ghosh, and Ashutosh Modi. 2024. IL-TUR: Benchmark for Indian legal text understanding and reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11460–11499, Bangkok, Thailand. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Saloni Sharma, Surabhi Srivastava, Pradeepika Verma, Anshul Verma, and Sachchida Nand Chaurasia. 2023. A comprehensive analysis of indian legal documents summarization techniques. *SN Comput. Sci.*, 4(5).

Abhay Shukla, Paheli Bhattacharya, Soham Poddar, Rajdeep Mukherjee, Kripabandhu Ghosh, Pawan Goyal, and Saptarshi Ghosh. 2022. Legal case document summarization: Extractive and abstractive methods and their evaluation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1048–1064, Online only. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 11328–11339. PMLR.

Hao Zheng and Mirella Lapata. 2019. Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247, Florence, Italy. Association for Computational Linguistics.

Appendix

A Chunking Strategy: Detailed Methodology

Our chunking approach follows the methodology of Bhattacharya et al. (2021), inspired by Gidiotis and Tsoumakas (2020), which addresses a fundamental challenge: pre-trained models have input length limits shorter than legal documents, yet naively using the same reference summary for all chunks ignores their varying content.

A.1 Chunk-Specific Summary Generation

Given a training pair (d, s) where d is a document and s is its reference summary:

Step 1: Document Segmentation. Partition d into n chunks: $d = \{d_1, d_2, \ldots, d_n\}$.

Step 2: Sentence-Level Mapping. For each sentence s_j in the reference summary s, identify its most similar sentence in the document d:

$$\text{map}(s_j) = \arg\max_{d_k \in d} \text{sim}(s_j, d_k) \tag{1}$$

where $\text{sim}(\cdot, \cdot)$ is a sentence similarity measure (detailed below).

Step 3: Chunk-Specific Summary Construction. For each chunk d_i, construct its target summary s_i by aggregating all summary sentences mapped to sentences within that chunk:

$$s_i = \{s_j \in s \mid \text{map}(s_j) \in d_i\} \tag{2}$$

This procedure generates multiple training pairs (d_i, s_i) from each document, where each chunk is paired with a semantically relevant subset of the original summary.

A.2 Similarity Measures

We compare two sentence similarity metrics:

A.2.1 TF-IDF Cosine Similarity (Baseline)

Representation. Sentences are represented as sparse TF-IDF vectors. We apply preprocessing (tokenization, stopword removal including legal terms, stemming) and vectorize using:

- 3,000-dimensional vocabulary
- Unigram + bigram features
- Sublinear TF scaling: $\text{tf}(t, d) = 1 + \log(\text{count}(t, d))$

Similarity. For sentences s_j (summary) and d_k (document) with TF-IDF vectors $\mathbf{v}_s$ and $\mathbf{v}_d$:

$$\text{sim}_{\text{TF-IDF}}(s_j, d_k) = \frac{\mathbf{v}_s \cdot \mathbf{v}_d}{\|\mathbf{v}_s\| \|\mathbf{v}_d\|} \tag{3}$$

Chunking Parameters.

- Fixed sliding window: 400 words, 50-word overlap
- Similarity threshold: 0.1 (sentence s_j maps to d_k if similarity ≥ 0.1)

This lexical baseline captures term overlap but ignores semantic context.

A.2.2 Mean Cosine Similarity with SBERT (MCS)

Representation. We use `all-MiniLM-L6-v2` Sentence-BERT (Reimers and Gurevych, 2019) to encode sentences as 384-dimensional dense embeddings. Unlike TF-IDF, SBERT operates on raw text and captures contextual semantics.

Similarity. For embeddings $\mathbf{e}_s, \mathbf{e}_d \in \mathbb{R}^{384}$:

$$\text{sim}_{\text{MCS}}(s_j, d_k) = \frac{\mathbf{e}_s \cdot \mathbf{e}_d}{\|\mathbf{e}_s\|_2 \|\mathbf{e}_d\|_2} \tag{4}$$

Chunking Parameters.

- Semantic chunking: respects paragraph boundaries, 500-word target size
- Similarity threshold: 0.4 (raised to mitigate false positives from dense embeddings)
- Quality filters: compression ratio $\in [0.05, 0.4]$, minimum 40 words

The higher threshold and quality filters prioritize precision over recall, trading dataset size for semantic coherence.

A.2.3 DSPy Prompt Configuration

The optimized prompt used in our hybrid extractive-abstractive pipeline:

Instructions: Create an extractive summary of a legal judgment by identifying and concatenating key paragraphs.

Important Notes:

- If example summaries are shown above, they are ONLY for demonstrating format and style
- Do NOT use factual content, names, or legal arguments from examples
- ONLY summarize the specific judgment provided in the "Judgment Text" field
- Each judgment is separate - do not mix information between cases

Instructions:

1. Identify key paragraphs covering: case parties/background, core legal issue, court's decision, and key reasoning
2. Optimize for ROUGE-2, ROUGE-L and BLEU by copying text as exactly as possible
3. Join paragraphs in logical order with minimal transitions
4. Preserve original legal terminology, case citations, and phrasing

Target: 500-700 words from current judgment only

Prompt Structure:

- **Input:** Full judgment text preceded by "— NEW JUDGMENT —"
- **Reasoning:** Chain-of-thought with "Let's think step by step in order to"
- **Output:** Extractive summary preceded by "— SUMMARY OUTPUT —"