A Budget Recipe for Finetuning a Long-form Legal Summarization Model

Chompakorn Chaksangchaichot[*]
VISAI AI, Thailand
chompakornc_pro@vistec.ac.th

Pawitsapak Akarajaradwong[*]
VISAI AI, Thailand
pawitsapaka_visai@vistec.ac.th

Abstract

We describe an inexpensive system that ranked first in the JUST-NLP 2025 L-SUMM task, summarizing very long Indian court judgments (up to 857k characters) using a single 80GB GPU and a total budget of about $50. Our pipeline first filters out length–summary outliers, then applies two-stage LoRA SFT on Qwen3-4B-Instruct-2507 to learn style and extend context, and finally runs RLVR tuned to BLEU, ROUGE-2, and ROUGE-L, with BLEU upweighted. We showed that two-stage SFT is better than a single-stage run, and RLVR gives the largest gains, reaching 32.71 internal vs. 16.15 base and 29.91 on the test leaderboard. In ablation on prompting, we find that a simple, naive prompt converges faster but saturates earlier, while the curated legal-structured prompt keeps improving with longer training and yields higher final scores, and the finetuned model remains fairly robust to unseen prompts. Our code[1] and models[2] are fully open-sourced, available for reproducibility.

1 Introduction

Summarization is one of the challenging tasks in the Natural Language Processing (NLP) domain (Zhang et al., 2024). Several benchmarks tried to measure different NLP systems using standard metrics n-gram-based metrics (Lin, 2004; Papineni et al., 2002) or LLM-as-judges (Li et al., 2024). In some specific case, summarization can be viewed as a domain-specific problem. For example, medical note summarization (Michalopoulos et al., 2022) in medical domains, meeting summary which usually summarizes key points and agenda (Kirstein et al., 2024), or in legal where summarization was done in legal cases (Datta et al., 2023; Shukla et al., 2022) or court judgments (Sharma et al., 2023). In this work, we focus specifically on legal judgment summarization, which is part of the JUST-NLP 2025 Legal Summarization (L-SUMM) Shared Task. We framed L-SUMM as a long-form summarization task where the length of the judgment can be extremely long (up to 857,477 characters).

To tackle this challenge, we utilized a standard finetuning technique followed by Reinforcement Learning with Verifiable Reward (RLVR). Additionally, we also constraint our compute budget to only $50 in GPU hours and only require a single GPU. To achieve such training efficiency, we utilized supervised finetuning using LoRA (Hu et al., 2021) on Qwen3-4B-Instruct-2507 (Yang et al., 2025). Our pipeline consists of three steps, two-stage supervised finetuning followed by a final RLVR for aligning LLM towards better summarization style. Our results showed that a two-stage supervised finetuning is necessary given a constrained compute budget compared to unified single stage. Furthermore, RLVR significantly boosted the finetuned model performance across all summarization metrics. However, we also noticed the instability in applying RLVR towards legal summarization across long-form summary. In our ablation study, we also investigate the effect of prompting towards downstream performance during supervised finetuning stage. In summary, we conclude our contributions as follows:

1. We provides a detailed system description of our design that ranked first place in the L-SUMM Shared Task.
2. We demonstrate an efficient training pipeline that achieves competitive summarization performance under a strict compute budget roughtly $50 using a single 80GB GPU.
3. We study the effect of prompting supervised finetuning data on legal summarization and report findings.

[*]These authors contributed equally as co-first authors.
[1]https://github.com/tann9949/justnlp-2025-legal-summ
[2]https://huggingface.co/VISAI-AI/Qwen3-4B-Instruct-2507-L-SUMM-fourcorners-rl-r2-ckpt500

2 Related Works

Reinforcement Learning with Verifiable Reward Reinforcement Learning with Verifiable Rewards (RLVR) was introduced as a more scalable alternative to the classical RLHF setup (Ouyang et al., 2022), which typically relies on PPO (Schulman et al., 2017), a critic head, and a learned reward model components that make RLHF costly to train. To reduce this overhead, Direct Preference Optimization (DPO) (Rafailov et al., 2024) became popular, but it still requires human preference pairs. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) addressed this by replacing preference data with rule-based, verifiable rewards, thereby removing the need for both a critic head and curated preference pairs. Subsequent works such as Dr. GRPO (Liu et al., 2025b) and DAPO (Yu et al., 2025) improved GRPO's sampling efficiency, reduced length bias, and refined clipping strategies. Group Sequence Policy Optimization (GSPO) (Zheng et al., 2025) further stabilized training for MoE models, by performing advantage renormalization at the sequence level instead of per token.

In this work, we adopt this RLVR to further align an instruction-tuned model for legal summarization, using verifiable task rewards. Prior studies have also shown that n-gram–based rewards (Chang et al., 2025) and semantic-similarity rewards (Akarajaradwong et al., 2025) can be effectively plugged into this framework to strengthen instruction following capabilities.

Parameter Efficient Approaches for Long Contextual Task Recent works proposed an efficient approach to conduct Parameter Efficient Finetuning (PEFT) towards long-form contents. LongLoRA (Chen et al., 2024) proposed using shifted-sparse attention to expand context length cheaply as well as it's quantized counterparts LongQLoRA (Yang, 2023). RST-LoRA (Liu and Demberg, 2024) adapts LoRA for long-document summarization by injecting RST discourse signals (centrality, relations, and their confidence) into the adapter, giving the model a richer, structure-aware fine-tuning signal and outperforming vanilla LoRA and full fine-tuning on multiple evaluations.

Legal Summarization There are several works introduced dataset for legal summarization task. Shukla et al. (2022) released three new case law summarization datasets drawn from Indian and UK Supreme Court judgments, and conducted extensive evaluations of extractive vs abstractive summarization methods on these legal cases. Building on this, Datta et al. (2023) presented the MILDSum corpus, a benchmark of 3,122 Indian legal judgments with high-quality summaries in both English and Hindi. Joshi et al. (2024) introduced IL-TUR, a broad benchmark for Indian Legal Text Understanding that includes a summarization task among eight diverse legal NLP tasks. IL-TUR provides multilingual legal datasets (spanning English and 9 Indian languages) and reports baseline results for each task.

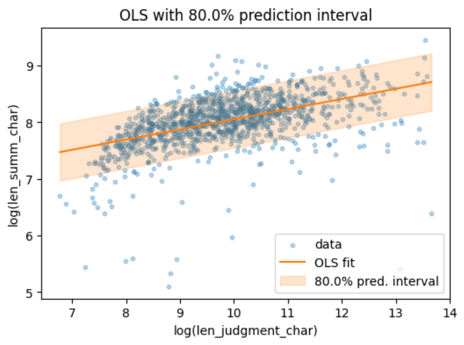

Figure 1: Line best fit and 80% prediction interval between judgment word counts and summary word counts in log-scale.

3 Methodology & Experimental Setups

Our system includes a data preparation phase that removes potential data outliers, followed by a two-stage instruction tuning pipeline before applying a final alignment tuning via RLVR.

3.1 Data Preparation

Upon inspecting the dataset, we found that the ratio between the judgment length and its summarized reference was sometimes disproportionate. In particular, some summaries were longer than the judgment, while others were shorter than expected.

To reduce noise before finetuning, we removed samples whose summary–judgment length ratio deviated from the expected trend. We first fitted a linear regression between judgment word counts and summary word counts in log-scale, then discarded any sample falling outside the 80% prediction interval, as illustrated in Figure 1. This filtering reduced the training set from 1,200 to 1,052 samples.

After filtering, we stratified the remaining data into training and internal validation splits, ensuring similar distributions of summary lengths in both. We used an 80/20 split, resulting in 841 training samples and 211 internal validation samples. Note that the official validation and test splits were kept held-out and only used for evaluation, as their reference summaries were not released.

3.2 Finetuning Pipeline

Our finetuning pipeline consists of two main stages, two-step supervised finetuning, followed by RLVR. The supervised stages adapt the base model to the task-specific summarization style, and the reinforcement step further aligns the model toward better word choice and target metrics via GSPO (Zheng et al., 2025). We use `Qwen/Qwen3-4B-Instruct-2507` (Yang et al., 2025) as the base model because of its strong instruction following capability and long context window (up to 262,144 tokens). We chose the non-thinking variant to reduce rollout cost during RL. All finetuning runs used LoRA (Hu et al., 2021) to stay within our compute budget. Training was constrained to a single A100 80GB GPU, costing about \$1.39/hour.[3]

Two-Stage Supervised Finetuning Because inputs can be very long and compute is limited, we split supervised finetuning into two stages.

In the first stage, we finetuned with a max sequence length of 16,384 tokens. We applied Rank-stabilized LoRA (Kalajdzievski, 2023) to all modules[4] using LoRA rank 256 and alpha of 32. We used Adam (Kingma and Ba, 2017) with a constant learning rate of 2e-4 following (Schulman and Lab, 2025), batch size 32, and trained for two epochs. This model is reported as 'SFT Stage 1'.

In the second stage, we merged the stage-1 adapter into the base model and continued training on longer-context samples. We filtered data to those with total prompt+completion length between 10,000 and 30,000 tokens, solely improve performance towards long input summary. To fit on a single 80GB GPU at this length, we reduced the LoRA rank to 32 and attached adapters only to `up_proj` and `down_proj`. We trained for one epoch with the same learning rate and batch size as stage 1. This model is reported as 'SFT Stage 2'.

Reinforcement Tuning Finally, we applied reinforcement tuning to directly optimize summarization metrics: BLEU, ROUGE-L, and ROUGE-2. BLEU was weighted 3× higher because the SFT model already achieved strong ROUGE but comparatively lower BLEU. We used DAPO (Yu et al., 2025) with high clip value and improved rollout sampling to increase reward diversity, and we computed importance sampling at sequence level following (Zheng et al., 2025). For optimizer, we used Adam 8-bit (Dettmers et al., 2022) with constant learning rate 8e-5, max grad norm 0.2, 16 rollouts, LoRA rank 2 and alpha 1, rollout temperature 1.0, max sequence length 12,000, and clip value 0.28 as in (Yu et al., 2025). We use low LoRA rank for following (Schulman and Lab, 2025) which stated that RLVR have a sparse reward thus lower training signal, making lower LoRA rank more suitable. We used the Unsloth (Daniel Han and team, 2023) implementation for efficient RL. During RL, we observed training instabilities (see Section 4), so selected the last stable checkpoint (step 500) for evaluation. This model is reported as 'Reinforcement Tuned' in the result table.

[3]Price according to https://www.runpod.io/pricing.

[4]`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`.

Model	Avg	Rouge-2	Rouge-L	BLEU
Internal Validation results				
Qwen3-4B-Thinking-2507	11.60	12.87	16.79	5.14
Qwen3-4B-Instruct-2507	16.15	18.79	22.31	7.33
SFT Stage 1	22.96	29.32	29.19	10.38
SFT Stage 2	24.55	30.89	30.49	12.29
Reinforcement Tuned	**32.71**	**35.60**	**34.43**	**28.11**
Validation Leaderboard results				
SFT Stage 1	25.47	31.25	31.42	13.74
SFT Stage 2	**25.57**	**31.51**	**31.77**	**13.43**
Test Leaderboard results				
SFT Stage 2	23.94	30.35	30.19	11.27
Reinforcement Tuned	**29.91**	**34.91**	**33.34**	**21.49**

Table 1: Performance of models across internal validation, validation leaderboard, and test leaderboard results.

4 Results & Discussion

The main results are summarized in Table 1.

Qwen3-4B Instruct outperforms the Thinking variant. From Table 1, the instruction-tuned model scores clearly higher than the thinking version on summarization. We attribute this to the nature of the task: judgments are long and information-dense, so additional chain-of-thought style reasoning may introduce token overhead and distract from concise summary generation. This observation supports our choice of `Qwen3-4B-Instruct-2507` as the base model.

Reinforcement tuning can collapse late in training. During GSPO-based reinforcement tuning, we observed occasional reward collapse, characterized by repetitive rollouts reaching the max se-

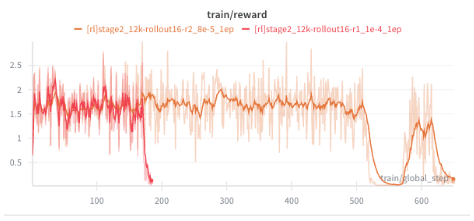

Figure 2: An observed behavior of reward collapsed when the model underwent GSPO training.

Model	Avg	Rouge-2	Rouge-L	BLEU
Internal Validation results				
SFT Stage 2	24.55	30.89	30.49	12.29
SFT Unified	22.67	29.45	29.07	9.49
Test Leaderboard results				
SFT Stage 2	27.21	33.36	32.25	16.01
SFT Unified	21.62	28.46	28.42	7.97

Table 2: Performance of the model that underwent two-stage supervised finetuning pipeline compared to unified pipeline.

quence length and a subsequent sharp drop in rewards as shown in Figure 2. We hypothesized that this collapse could resulted from an instability inference-training mismatch, which can be either from precision sensitivity (Qi et al., 2025), or hardware instability (Liu et al., 2025a). To avoid evaluating such degraded checkpoints, we selected the last stable checkpoint (step 500) for reporting. We also provide some analysis of model performance on different checkpoints on Appendix B.

Cost analysis of the best model. Our full pipeline used about 35 GPU-hours in total[5] On RunPod pricing ($1.39/hour), this corresponds to roughly $50 for end-to-end training, making the approach practical for single-GPU setups.

5 Ablation Study

To further validate the effectiveness of our approach, we conducted two ablation studies. First, we examined whether the two-stage supervised finetuning pipeline can be replaced with a single unified stage. Second, we evaluated the impact of prompt design by comparing a carefully curated prompt against a naive prompt (see Appendix A).

Two-stage supervised finetuning outperforms a unified pipeline. Because our finetuning strategy attaches different LoRA adapters across two stages, we tested whether training only with the second-stage setup (long-context, LoRA only on MLP modules) could match its performance. To do this, we finetuned `Qwen3-4B-Instruct-2507` using the stage-2 hyperparameters, increased the training epochs to 3, and removed the minimum prompt-length filter. We denote this as 'SFT Unified' in Table 2. As shown in the table, **SFT Stage 2 consistently outperforms SFT Unified**, highlighting the benefit of first adapting the model on shorter contexts with more trainable parameters before extending to long-context finetuning.

Naive prompts converge faster, but curated prompts win with longer training. We also compared two-stage finetuning under two prompts: a naive prompt with no heuristic guidance (Appendix A) and our curated summarization prompt. Both models used the same stage-1 finetuning setup described in Section 3.2. The results are summarized in Figure 3.

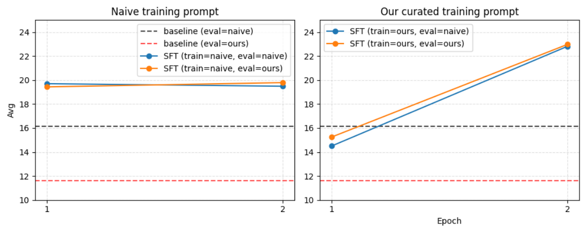

Figure 3: Plot of internal validation performance of Qwen3-4B-Instruct-2507 underwent SFT under different prompt, and evaluate on different prompt. Horizontal axis denotes training epochs and vertical axis denotes average metric among ROUGE-2, ROUGE-L, and BLEU.

We observed that the naive prompt learns faster in the early steps, reaching an average score of ∼19.5 after the first epoch, while the curated prompt lags behind at ∼14.5, slightly below baseline. However, after the second epoch, the naive-prompt model saturates and gains little, whereas the model trained with the curated prompt continues to improve beyond an average score of 22. This suggests that **well-designed summarization prompts yield better final performance when training runs longer**.

We also tested cross-prompt robustness by evaluating models on prompts different from those seen in training. Under our setup and hyperparameters, **the finetuned models did not exhibit strong sensitivity to unseen prompts**, indicating reasonable prompt generalization.

[5]51 minutes for SFT stage 1, 31 minutes for SFT stage 2, and 33 hours for reinforcement tuning.

6 Conclusion

We presented a practical pipeline for finetuning `Qwen3-4B-Instruct-2507` on long-form legal summarization. Our approach begins with length-based data filtering to remove samples whose judgment–summary ratio deviates from the expected trend, reducing noise in the training set. We then apply a two-stage supervised finetuning strategy with LoRA, first adapting the model on shorter contexts and then extending to long-context data, followed by GSPO-based reinforcement tuning to directly optimize ROUGE and BLEU. Experiments showed that the two-stage SFT setup outperforms both the base model and a unified SFT pipeline, while RL yields the largest performance gains on internal and leaderboard evaluations. Our ablations further suggest that prompt design matters more at longer training horizons and that the model remains reasonably robust to unseen prompts. Notably, the entire recipe fits within a single 80GB GPU and costs roughly $50, demonstrating that competitive legal summarization is achievable under modest compute constraints.

Limitations

This work has several limitations. First, our evaluation of the two-stage finetuning pipeline is incomplete: we fixed the second-stage training to a single epoch and did not systematically explore longer training, curriculum-style schedules, or error breakdowns by sequence length. As a result, it is still unclear whether additional long-context finetuning would yield further gains or help specific length regimes.

Second, the reinforcement tuning phase exhibited training collapse, and we only mitigated it operationally (by selecting the last stable checkpoint) rather than fully diagnosing or fixing the underlying cause. A deeper study of rollout diversity, reward shaping, and clipping strategies would likely improve stability.

Third, we did not exhaust the RLVR design space: we did not compare against vanilla GRPO (Shao et al., 2024) without sequence-level grouping, nor did we test alternative reward configurations such as single-metric rewards versus our compound setup. These choices may affect both final quality and robustness.

Finally, the reported $50 training cost reflects only the successful run on a single 80GB GPU. When including exploratory and failed runs, the total compute was closer to 70 GPU-hours (about $100), so the true end-to-end cost of reproducing this work may be higher than the headline number.

References

Pawitsapak Akarajaradwong, Chompakorn Chaksangchaichot, Pirat Pothavorn, Attapol Thamrongrattanarit-Rutherford, Ekapol Chuangsuwanich, and Sarana Nutanong. 2025. Can group relative policy optimization improve thai legal reasoning and question answering? *Preprint*, arXiv:2507.09638.

Yapei Chang, Yekyung Kim, Michael Krumdick, Amir Zadeh, Chuan Li, Chris Tanner, and Mohit Iyyer. 2025. Bleuberi: Bleu is a surprisingly effective reward for instruction following. *Preprint*, arXiv:2505.11080.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. *Preprint*, arXiv:2309.12307.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

Debtanu Datta, Shubham Soni, Rajdeep Mukherjee, and Saptarshi Ghosh. 2023. MILDSum: A novel benchmark dataset for multilingual summarization of Indian legal case judgments. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5291–5302, Singapore. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2022. 8-bit optimizers via block-wise quantization. *Preprint*, arXiv:2110.02861.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

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.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *Preprint*, arXiv:2312.03732.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Frederic Kirstein, Terry Ruas, Robert Kratel, and Bela Gipp. 2024. Tell me what I need to know: Exploring

LLM-based (personalized) abstractive multi-source meeting summarization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 920–939, Miami, Florida, US. Association for Computational Linguistics.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *Preprint*, arXiv:2412.05579.

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

Dongqi Liu and Vera Demberg. 2024. RST-LoRA: A discourse-aware low-rank adaptation for long document abstractive summarization. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2200–2220, Mexico City, Mexico. Association for Computational Linguistics.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. 2025a. When speed kills stability: Demystifying rl collapse from the inference-training mismatch.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

George Michalopoulos, Kyle Williams, Gagandeep Singh, and Thomas Lin. 2022. MedicalSum: A guided clinical abstractive summarization model for generating medical reports from patient-doctor conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4741–4749, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Defeating the training-inference mismatch via fp16. *Preprint*, arXiv:2510.26788.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

John Schulman and Thinking Machines Lab. 2025. Lora without regret. *Thinking Machines Lab: Connectionism*. Https://thinkingmachines.ai/blog/lora/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jianxin Yang. 2023. Longqlora: Efficient and effective method to extend context length of large language models. *Preprint*, arXiv:2311.04879.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Haopeng Zhang, Philip S. Yu, and Jiawei Zhang. 2024. A systematic survey of text summarization: From statistical methods to large language models. *Preprint*, arXiv:2406.11289.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

A Prompts

The system prompt was outlined as follows:

System Prompt

You are a precise summarizer for legal materials. Follow the user "Instruction" steps exactly.

For instruction prompt, we sampled different summary item from the training sample and extract key patterns found in the summarization. The prompt is outlined below.

Instruction Prompt

Instruction
You are given an Indian court judgment in English to summarize. Your task is to summarize the judgment based on the following strategies on summarizing the judgment:
1) Identify the authority (court, commission, regulator, ministry or parliamentary body) and its action in the opening clause – name the institution and state what it did (granted/-denied/quashed/stayed/notified/imposed, etc.), adding the date if given.
2) Summarise the parties and the core issue in one sentence – who is involved (petitioner, accused, complainant, regulator) and what the underlying dispute/offence/claim is, keeping background facts concise.
3) Mention the legal basis or tests only when they appear in the source – specify Acts, Sections, Articles, Rules or Regulations exactly as provided; never invent citations if none are given.
4) Convey the key reasoning and observations succinctly – what the bench or adjudicating authority held/observed/noted and why, paraphrasing where possible and using short quoted phrases only when present.
5) Finish with the outcome and operative directions – note the practical effect (e.g. bail granted/refused, penalty imposed, matter stayed), any deadlines, next-hearing dates or compliance steps. If the source lists multiple directions, collapse them into one sentence separated by semicolons rather than using bullets or separate lines.

Style & constraints:
- Neutral, factual tone in past tense and active voice.
- Prefers in paragraphic format not in bullet points.
- Do not include headings, bullet points or commentary.
- Preserve names, numbers and dates exactly; if a detail is absent in the source, simply omit it rather than guessing.

Input Judgment
`{{judgment}}`

Output Summary:

We also provide our naive summary prompt here.

Naive Summary Prompt

Summarize the following judgement:

Input Judgment:
`{{judgment}}`

Output Summary:

B Performance of Models Under Reinforcement Tuning Across Training Steps

In Section 4, we discussed that the reinforcement tuning stage sometimes exhibited a *collapse* of the reward signal as training progressed. To better understand this phenomenon, we conducted two runs that differed only in LoRA rank and learning rate.

The first run (denoted as `GSPO_r1_1e-4` in Table 3) followed the hyperparameters in Section 3.2, except that we used a LoRA rank of 1 and a learning rate of 1e-4. The second run (our best-performing configuration), denoted as `GSPO_r2_8e-5`, followed the settings in Section 3.2 with a higher LoRA rank 2 and a lower learning rate 8e-5.

In Figure 2, which plots the reward over training steps, `GSPO_r1_1e-4` corresponds to the red line, while `GSPO_r2_8e-5` corresponds to the orange line. We observe that `GSPO_r1_1e-4` collapses relatively early, around checkpoint 190. To improve stability, we therefore reduced the learning rate and increased the LoRA rank. This second run was indeed more stable, but it still eventually collapsed at around the 520th step.

To check whether the model's *task performance* actually improved before the collapse, we saved a checkpoint every 50 steps and evaluated each checkpoint on our internal validation split. The results are shown in Table 3.

Model	**Ckpt**	**Avg**	**ROUGE-2**	**ROUGE-L**	**BLEU**
Internal validation results					
`GSPO_r1_1e-4`	50	26.29	32.15	31.64	15.09
`GSPO_r1_1e-4`	100	28.60	33.79	32.53	19.47
`GSPO_r1_1e-4`	150	28.36	33.95	32.89	18.26
`GSPO_r2_8e-5`	300	31.50	35.59	33.99	24.92
`GSPO_r2_8e-5`	400	31.11	35.75	33.97	23.61
`GSPO_r2_8e-5`	500	32.71	35.60	34.43	28.11
Test leaderboard results					
`GSPO_r1_1e-4`	150	27.21	33.36	32.25	16.01
`GSPO_r2_8e-5`	500	29.91	34.91	33.34	21.49

Table 3: Internal validation and test leaderboard performance across different reinforcement tuning steps. "Avg" is the average of the reported metrics.

Overall, we observe an upward trend in downstream metrics as training progresses, despite the reward curve eventually collapsing. For example, in the more stable run (`GSPO_r2_8e-5`), BLEU increases from 24.92 at step 300 to 28.11 at step 500 on the internal validation split, and the corresponding test leaderboard score at step 500 is 21.49. At the same time, ROUGE-2 and ROUGE-L either improve slightly or remain in a similar range across checkpoints. These results suggest that the apparent instability in the reward signal is not primarily caused by degraded task learning, but is more consistent with other factors (e.g., inference–training mismatch) discussed in Section 4.