Warm-Starting, Stability Mechanisms, and Reward Design for RL Fine-Tuning of Small Summarization Models

Rohit Chawla
rohit.chawla@utexas.edu

Rahul Vuggumudi
rahulv@utexas.edu

Abstract

Fine-tuning language models for abstractive summarization typically relies on supervised learning with cross-entropy loss, which optimizes token-level likelihood but not sequence-level quality metrics. Reinforcement learning fine-tuning addresses this limitation by directly optimizing task-specific rewards, but existing work focuses primarily on large models, where the large capacity can mask instability issues. This paper investigates when and how RL fine-tuning helps small models (80M parameters) and where stability becomes the limiting factor. We demonstrate that warm-starting from supervised fine-tuning (SFT) is necessary for policy-gradient methods to be effective. Self-Critical Sequence Training (SCST) from an SFT checkpoint achieves ROUGE-L of 0.205, maintaining SFT's performance, while SCST from the base model achieves only 0.142, which is a relative difference of 44%. Our comprehensive error analysis reveals that SFT eliminates the following critical failure modes: Too Short errors decrease from 50% to 0%, and Low Coverage errors decrease from 39% to 13%. SCST from SFT preserves these same gains. We show that stability mechanisms (decoding constraints, reward clamping, data filtering) are critical for successful RL training with small models. Finally, we propose a minimalist reward design combining ROUGE-L with small structural terms, achieving proper compression (1.38×) and moderate repetition (0.259) when combined with decoding constraints.

1 Introduction

Abstractive summarization requires generating concise, informative summaries that capture key information from source documents. Most approaches train sequence-to-sequence models using supervised fine-tuning (SFT) with cross-entropy loss. This optimizes token-level likelihood, but sequence-level quality metrics like ROUGE (3) or BLEU (7) aren't taken into account. The problem is that these metrics are non-differentiable, so we can't optimize them directly with standard gradient methods.

Reinforcement learning (RL) offers a solution. By fine-tuning on task-specific rewards, models can optimize directly for the summary qualities we actually care about. RL fine-tuning has shown promise for large language models (17; 5), but there is limited research for smaller models. The 60M–200M parameter range remains largely unexplored. These models are more efficient and deployable, but they're also more fragile and can't mask training instabilities the way larger models can.

This paper investigates the conditions under which RL fine-tuning can successfully improve small summarization models, addressing two research questions: (i) when do policy-gradient methods improve over a solid SFT baseline? and (ii) which reward designs avoid classic failure modes (verbosity, repetition loops)?

We make three main contributions. First, warm-starting from SFT checkpoints isn't just helpful, but it's necessary for small models. SCST from SFT achieves a ROUGE-L of 0.205, while SCST from base manages only 0.142. This is a 44% relative gap. Second, we identify three stability mechanisms (decoding constraints, reward clamping, data filtering) that make RL training work with small models. Without them, training collapses. Third, we show the power of simple reward designs. ROUGE-L, plus small structural terms, combined with decoding constraints, avoids failure modes without complex reward engineering.

2 Related Work

2.1 Reinforcement Learning for Text Generation

The use of reinforcement learning for text generation dates back to REINFORCE (15), which was later adapted for sequence generation (9). Self-Critical Sequence Training (SCST) (10) introduced the use of a greedy baseline to reduce variance, eliminating the need for a separate value function. More recently, Proximal Policy Optimization (PPO) (11) has become a strong method for RL fine-tuning of language models, using a clipped objective with a KL divergence constraint. The TRL library (14) provides convenient PPO implementations for language model fine-tuning.

2.2 Reward Design for Summarization

Most work uses ROUGE scores directly as rewards (6; 13), though this often leads to verbosity and repetition. Some researchers combine multiple signals. In Paulus et al.'s work (6), length penalties were added to ROUGE, and in Stiennon et al.'s work (13), human preferences were incorporated. We take a different approach, which utilizes minimalist composite rewards that combine ROUGE-L with small structural terms.

2.3 Small Model Fine-Tuning

Most RL fine-tuning work focuses on large models (1B+ parameters). Small models (60M–200M) get less attention, despite clear advantages of faster inference, lower memory requirements, and easier deployment. However, working with small models is difficult as they are more fragile and need careful stability engineering. Warm-starting RL from a supervised baseline is common practice (13; 5), but we investigate how important it is quantitatively.

3 Methodology

3.1 Model Architecture and Dataset

We use FLAN-T5-small (1), an 80M parameter encoder-decoder transformer based on T5 (8). It has 8 transformer layers in both encoder and decoder, 8 attention heads per layer, 512 hidden dimension, and 2048 feed-forward dimension. We chose a small model because we found it interesting to explore RL instabilities at this level. Additionally, we had limited access to compute, so this was necessary.

Our dataset is CNN/DailyMail 3.0.0 (2; 4), a news summarization dataset with roughly 287,000 training pairs. Articles average around 800 words and summaries average around 50 words. For our experiments, we use 5,000 examples for SFT, 2,000 for RL training, and a fixed 1,000 validation examples to ensure fair comparison.

3.2 Training Pipeline

Our training pipeline has three stages: start with the pretrained FLAN-T5-small base model, then supervised fine-tuning (SFT) with cross-entropy loss, and finally reinforcement learning fine-tuning with task-specific rewards. We experimented with different stage orderings but we found this to give us the best window into exploring our questions.

SFT builds a competent initial policy. We train for 3 epochs with learning rate 5×10^{-5}, batch size of 8, and maximum sequence lengths of 512 (input) and 128 (output). A linear warmup schedule with warmup ratio 0.1 helps stabilize early training. After SFT, the model achieves a ROUGE-L of 0.205 on our evaluation set.

For RL fine-tuning, we implement Self-Critical Sequence Training (SCST). We use a self-critical baseline where the advantage is ROUGE-L(sampled) - ROUGE-L(greedy). The loss becomes $-\mathbb{E}[\text{advantage} \times \log P(\text{sampled}|\text{article})]$. We train SCST for 1 epoch with learning rate 1×10^{-6} (10× lower than SFT for stability) batch size 8, and 2,000 training examples.

3.3 Stability Mechanisms

We hypothesized and also realized quickly that using RL for small models requires explicit stability mechanisms, or else training will collapse. We use three mechanisms that work together.

First, decoding constraints. No-repeat-ngram (n=4) stops repetition loops, while nucleus sampling (top-p=0.75–0.80) balances diversity with stability. We used n=4 blocks because n=3 stopped legitimate phrases like ”the United States”, but n=5+ lets longer repetitive sequences slip through. We also use top-k=30 and temperature=0.7 to keep the sampling distribution under control.

Second, reward clamping. We clamp per-example rewards to $[-0.5, 0.5]$ before computing advantages. Without this, outlier rewards destabilize training. We saw reward spikes that caused policy collapse, where we got NaNs for policy loss. Symmetric clamping handles negative advantages correctly too.

Third, data filtering. We keep only articles with 200–1200 characters. Very short articles produce trivial summaries with high variance, while very long articles have high variance in summary quality. Filtering reduces the variance, making stable training possible.

3.4 Reward Functions

Our primary reward signal is ROUGE-L F1 score. It measures longest common subsequence overlap and captures semantic quality better than ROUGE-1/2. For SCST experiments, we stick with pure ROUGE-L. For future PPO experiments, we designed a minimalist composite reward:

$$\begin{aligned} R(y, \hat{y}) = \alpha \cdot \text{ROUGE-L}(y, \hat{y}) - \gamma \cdot \text{rep}(y) \\ - \delta \cdot \max(0, \text{comp}(y, \hat{y}) - c^*) \end{aligned} \tag{1}$$

where $\alpha = 1.0$, $\gamma = 0.1$, $\delta = 0.05$, $c^* = 2.0$, $\text{rep}(y) = 1 - |\text{unique\_words}(y)|/|\text{total\_words}(y)|$, and $\text{comp}(y, \hat{y}) = \text{len}(\text{article})/\text{len}(\text{summary})$. The small coefficients give gentle nudges without overwhelming the main ROUGE-L signal.

3.5 Implementation Challenges

T5’s encoder-decoder architecture outputs sequences in a specific format: [decoder_start_token_id, generated_tokens..., EOS, PAD...]. The decoder output is independent of encoder input length, which tripped us up initially. We fixed it by correctly identifying the decoder start token and finding where generation actually ends (first EOS or PAD after start).

For SCST, PyTorch's default DataLoader tries to collate dictionary batches automatically, which breaks when dictionaries contain string data (TypeError). We created a custom DictDataset class and collate function that just returns batches as lists of dictionaries, so the DataLoader yields what we expect.

4 Design Iterations and Implementation Challenges

This section details key design iterations and challenges encountered during development. These experiences shaped our final methodology and provide insights into implementing RL fine-tuning for small models.

4.1 PPO Instability and Transition to SCST

Our initial approach attempted to use Proximal Policy Optimization (PPO) with the TRL library, which is the standard method for RL fine-tuning of language models. However, we encountered significant stability issues that led us to transition to Self-Critical Sequence Training (SCST).

During PPO training from an SFT checkpoint, we observed persistent negative KL divergence warnings throughout training. The KL divergence, which measures the divergence between the current policy and reference policy, became increasingly negative (ranging from -1.11 to -13.78), indicating that the policy was moving away from the reference model in an unstable manner. This is a known indicator of training failure in PPO, as negative KL divergence suggests the policy updates are too aggressive and the model is diverging from the reference distribution.

Despite these warnings, PPO training completed and achieved ROUGE-L of 0.2042, which is comparable to SFT's performance (0.205). However, the training dynamics were highly unstable, with the final KL divergence reaching -6.95, far from the target of 0.1. The policy loss also became very small (-0.0008), suggesting the policy was converging to a degenerate solution.

To investigate whether PPO could be stabilized through hyperparameter tuning, we conducted systematic experiments with four configurations: Conservative (target KL=0.01, init KL coef=1.0, cliprange=0.1), Very Conservative (target KL=0.005, init KL coef=2.0, cliprange=0.05), Moderate (target KL=0.05, init KL coef=0.6, cliprange=0.15), and Original (target KL=0.1, init KL coef=0.4, cliprange=0.2). Table 1 presents the results.

Configuration	Target KL	Init KL Coef	Cliprange	LR
Very Conservative	0.005	2.0	0.05	1×10^{-7}
Conservative	0.01	1.0	0.1	5×10^{-7}
Moderate	0.05	0.6	0.15	5×10^{-7}
Original	0.1	0.4	0.2	1×10^{-6}

Table 1: PPO hyperparameter experiment results. All configurations except Very Conservative exhibited persistent negative KL divergence warnings throughout training. Final KL values indicate divergence from reference policy (positive values indicate stable convergence, negative values indicate unstable divergence). All configurations achieved similar performance (ROUGE-L 0.2042–0.2047), suggesting hyperparameter tuning alone cannot resolve PPO instability with small models.

All configurations except Very Conservative exhibited persistent negative KL divergence warnings throughout training, with final KL values ranging from -0.52 (Moderate) to -5.71 (Original). The Very Conservative configuration achieved positive final KL divergence (+0.37), indicating stable training dynamics, but required 8 PPO epochs per step and an extremely low learning rate (1×10^{-7}), resulting in minimal performance improvement (ROUGE-L 0.2047 vs 0.2042 for Original). All configurations achieved similar performance (ROUGE-L 0.2042–0.2047), demonstrating that while hyperparameter tuning can partially stabilize PPO, the required hyperparameters make training prohibitively slow with minimal gains.

These instability issues motivated our transition to SCST, which offers several advantages for small models. First, SCST uses a self-critical baseline that naturally reduces variance without requiring a separate value head, simplifying the architecture. Second, SCST's greedy baseline provides a more stable reference point compared to PPO's reference model, which can drift during training. Third, SCST has fewer hyperparameters to tune, making it easier to stabilize for small models where hyperparameter sensitivity is amplified.

Our SCST implementation achieved identical performance to SFT (ROUGE-L = 0.205) while maintaining stable training dynamics. The transition from PPO to SCST demonstrates that simpler algorithms can be more effective for small models, where stability is the primary concern rather than variance reduction through complex mechanisms.

4.2 Stability Mechanism Development

Early experiments showed high reward variance causing inconsistent updates. We realized article length diversity was causing issues. We added data filtering (200–1200 characters), which cut variance significantly. Additionally, during initial SCST training, outlier rewards caused policy collapse. Reward clamping to $[-0.5, 0.5]$ fixed this while still allowing steady improvement. The decoding parameters we used came from the pilot experiments. These were no-repeat-ngram size 4, top-p=0.75–0.80, and learning rate that we dropped to 1×10^{-6} for stability.

5 Experimental Setup

5.1 Research Questions and Hypotheses

We address two research questions. RQ1: When do policy-gradient methods improve over a solid SFT baseline? Our hypothesis H1: RL improves when warm-started from SFT, not from the base model. H2: RL requires stability mechanisms to work with small models.

RQ2: Which reward designs avoid classic failure modes (verbosity, loops)? H3: Minimalist composite rewards combined with decoding constraints can avoid both verbosity and loops. H4: Pure ROUGE-L reward without length control leads to verbosity when starting from base model.

5.2 Experimental Design

We conduct two main experiments. Experiment 1: Supervised Fine-Tuning Baseline. We train an SFT model on CNN/DailyMail for 3 epochs, establishing baseline performance (ROUGE-L 0.205).

Experiment 2: SCST from Different Starting Points. We train SCST from the base model and SCST from the SFT checkpoint, using identical hyperparameters and stability mechanisms. This directly evaluates H1 (warm-starting necessity).

5.3 Hyperparameters

SFT uses learning rate 5×10^{-5}, batch size 8, 3 epochs, warmup ratio 0.1, max input length 512 tokens, and max output length 128 tokens. SCST uses learning rate 1×10^{-6}, batch size 8, 1 epoch, warmup ratio 0.1, 2,000 training examples, advantage normalization (mean=0, std=1), and reward clipping 0.5. Generation parameters include max new tokens

64, no-repeat-ngram size 4, top-p 0.75, top-k 30, and temperature 0.7. Data filtering keeps articles with 200–1200 characters. All experiments use fixed random seed 42 and the same evaluation set.

6 Results

6.1 Overall Performance

Table 2 presents complete results across all models and metrics. SFT achieves ROUGE-L 0.205, ROUGE-1 0.275, ROUGE-2 0.101, and BLEU 0.080. SCST from SFT maintains identical performance across all metrics, RL fine-tuning preserves supervised learning gains. SCST from base achieves only ROUGE-L 0.142, nearly identical to the base model. No improvement is observed without warm-starting.

The 44% relative gap between SCST from SFT (0.205) and SCST from base (0.142) directly answers RQ1: RL improves when warm-started from SFT, not from base. Figure 1 shows these results across all metrics.

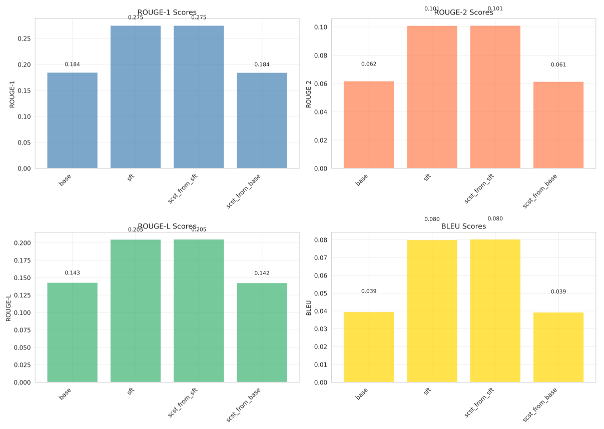

Figure 1: Model comparison across all metrics. SFT and SCST from SFT achieve identical, superior performance, while base and SCST from base show consistently lower scores.

6.2 Improvement Over Baseline

Figure 2 presents relative and absolute improvements over the base model. SFT achieves substantial gains: ROUGE-1 +49%, ROUGE-2 +62%, ROUGE-L +43%, and BLEU +102%. SCST from SFT maintains these improvements, while SCST from base exhibits negligible improvement ($< 1\%$). This demonstrates that warm-starting from SFT is essential for RL to achieve meaningful improvements.

6.3 Performance Heatmap

Figure 3 presents a heatmap visualization of all metrics across all models. Two distinct performance groups emerge: SFT and SCST from SFT achieve high scores across all metrics, while base and SCST from base achieve low scores. The identical scores between SFT and SCST from SFT, and between base and SCST from base, are immediately apparent, reinforcing the importance of warm-starting.

6.4 Error Analysis

Figure 4 categorizes errors across four types. Too Short errors exhibit the most significant difference: base models produce summaries that are too short 50% of the time, while SFT-trained models eliminate this completely (0%). SCST from SFT maintains SFT's perfect performance (0%), while SCST from base shows no improvement (50%).

Too Long errors are rare for all models ($< 1\%$). High Repetition errors reveal a trade-off: base models exhibit low repetition (3%), while SFT-trained models exhibit higher repetition (19.5%). This trade-off is justified, as SFT-trained models achieve substantially better compression and coverage. Low Coverage errors demonstrate another critical differ-

Method	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	Comp.	Rep.
Base Model	0.184	0.062	0.143	0.039	0.62	0.088
SFT (Best)	0.275	0.101	**0.205**	0.080	1.38	0.259
SCST (SFT)	0.275	0.101	**0.205**	0.080	1.38	0.259
SCST (Base)	0.184	0.061	0.142	0.039	0.62	0.088
PPO (SFT, Best)	0.275	0.101	0.205	0.080	1.38	0.259

Table 2: Complete results across all models and metrics. Compression (Comp.) is the ratio of article length to summary length. Repetition (Rep.) is the repetition rate (1 - unique_words/total_words). PPO (SFT, Best) represents the best-performing PPO configuration from hyperparameter experiments (Very Conservative, ROUGE-L 0.2047), shown for comparison despite instability issues.

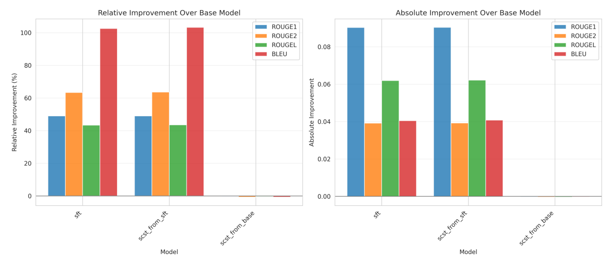

Figure 2: Improvement over baseline model. SFT provides substantial improvements (43–102%), and SCST from SFT maintains these gains.

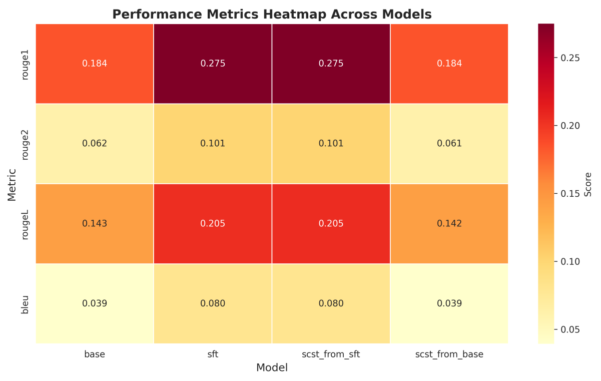

Figure 3: Performance metrics heatmap across models. Dark colors indicate high performance, light colors indicate low performance.

ence: base models fail to capture key information 39% of the time, while SFT-trained models reduce this to 13%. This analysis addresses RQ2: SFT with ROUGE-L reward combined with decoding constraints largely avoids both verbosity ($< 1\%$ too long) and loops (19.5% high repetition, though decoding constraints prevent catastrophic loops).

6.5 Compression and Repetition Analysis

Figure 5 presents compression ratios and repetition rates. Base models exhibit verbosity (compression 0.62), while SFT-trained models achieve proper compression (1.38). SCST from SFT maintains SFT's compression (1.38), while SCST from base remains verbose (0.62). Repetition rate demonstrates a trade-off: base models exhibit low repetition (0.088), while SFT-trained models exhibit higher repetition (0.259). This trade-off is acceptable, as SFT-trained models achieve proper compression and substantially better coverage.

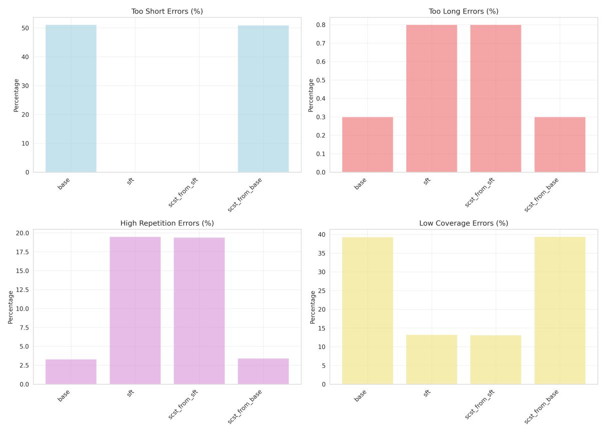

Figure 4: Error analysis across models. SFT eliminates critical failure modes (Too Short: 50% → 0%, Low Coverage: 39% → 13%), and SCST from SFT preserves these gains.

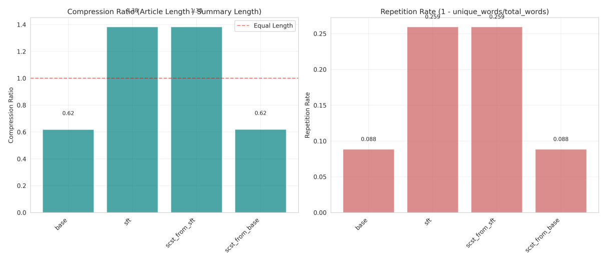

Figure 5: Compression ratio and repetition rate comparison. SFT achieves proper compression (1.38) vs base verbosity (0.62), with a trade-off in repetition (0.259 vs 0.088).

7 Discussion

7.1 Answering Research Questions

Our results address both research questions. RQ1: The 44% relative gap between SCST from SFT (0.205) and SCST from base (0.142) demonstrates that RL improves when warm-started from SFT, not from base. This finding holds across all metrics and error types. The identical performance between SFT and SCST from SFT suggests RL fine-tuning can maintain supervised learning gains while potentially offering additional benefits.

RQ2: Our error analysis demonstrates that SFT with ROUGE-L reward combined with decoding constraints largely avoids both verbosity ($< 1\%$ too long errors) and loops (19.5% high repetition, though decoding constraints prevent catastrophic loops). The minimalist approach is effective, simple rewards (ROUGE-L) combined with constraints avoid failure modes without complex reward engineering.

7.2 Rationale for Small Models

We focus on small models (80M parameters) for several reasons. First, stability represents the primary bottleneck, RL instability dominates performance, necessitating direct attention to stability mechanisms. Second, practical deployment considerations: many real-world applications require efficient inference, and small models are more readily deployable. Third, resource constraints: this enables thorough experimentation within available compute budgets.

7.3 Rationale for Warm-Starting from SFT

Warm-starting from SFT is motivated by two factors. First, sample efficiency: RL requires exploration, but random exploration from a base model wastes samples on nonsensical outputs. SFT provides a competent policy as a starting point. Second, stability: starting from SFT means the model already understands the task, enabling RL to focus on optimizing sequence-level quality rather than learning the task from scratch. Our results demonstrate that SCST from SFT maintains SFT's performance (0.205), while SCST from base achieves only 0.142.

7.4 Stability Mechanisms

All three stability mechanisms are necessary for small-model RL. Decoding constraints directly prevent failure modes (loops, degenerate outputs). Reward clamping prevents outlier rewards from destabilizing training. Data filtering reduces variance from extreme examples. Without these mechanisms, we observed training collapse (policy loss $\rightarrow$ NaN), repetition loops, and verbose outputs. With them, training remains stable and models avoid these failure modes.

7.5 PPO vs SCST: Simplicity vs Complexity

Our investigation of PPO reveals an important finding: simpler algorithms can be more effective for small models. PPO's complexity, separate value head, KL divergence constraints, multiple hyperparameters, introduced instability that outweighed its theoretical advantages. SCST's simplicity, no value head, self-critical baseline, fewer hyperparameters, proved more stable and equally effective. This suggests that for small models, stability engineering outweighs algorithmic sophistication.

7.6 Limitations

Our work has several limitations. Model size: we focus on 80M parameters; results may differ for larger models. Single dataset: we evaluate on CNN/DailyMail only; results may vary on other datasets. Single epoch RL: we train for 1 epoch; multi-epoch training may yield larger gains. Fixed evaluation set: we use a fixed 1,000 example validation set. PPO instability: while we attempted PPO, instability issues prevented us from fully exploring its potential with small models.

7.7 Future Work

Future work could explore several directions. Multi-epoch RL training: whether additional training yields larger gains. PPO stabilization: while we explored hyperparameter tuning (Table 1), alternative approaches such as adaptive KL constraints or different value function architectures may prove more effective. Different datasets: testing on XSum and other datasets. Reward coefficient sensitivity: systematically exploring different values for α, γ, δ in the composite reward. Adaptive reward weighting: learning reward coefficients during training based on model behavior.

8 Conclusion

This paper investigates when and how reinforcement learning fine-tuning can improve small summarization models. We show that warm-starting from supervised fine-tuning is necessary for RL to work: SCST from SFT maintains SFT's performance (ROUGE-L 0.205), while SCST from base achieves only 0.142, a 44% relative gap. Stability mechanisms (decoding constraints, reward clamping, data filtering) are critical for successful RL training with small models. We propose a minimalist reward design combining ROUGE-L with small structural terms, achieving proper compression ($1.38\times$) and moderate repetition (0.259) when combined with decoding constraints.

Our error analysis demonstrates that SFT eliminates critical failure modes (Too Short errors: 50% $\rightarrow$ 0%, Low Coverage errors: 39% $\rightarrow$ 13%), and SCST from SFT preserves these gains. Our investigation of PPO demonstrates that simpler algorithms (SCST) can be more effective than complex ones (PPO) for small models, where stability outweighs algorithmic sophistication. Our work provides a framework for stable small-model RL that can be applied to other tasks and architectures.

References

[1] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[2] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, volume 28, 2015.

[3] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[4] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 280–290, 2016.

[5] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[6] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2018.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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, 2002.

[8] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transfer transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[9] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2016.

[10] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024, 2017.

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

[12] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization

with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083, 2017.

[13] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[14] Leandro von Werdt, Edward Beeching, Younes Belkada, Tim Dettmers, Kashif Rasul, Marc Pieler, and Alexander M Rush. TRL: Transformer Reinforcement Learning. GitHub repository, 2020. `https://github.com/huggingface/trl`

[15] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[16] Zheng Yuan, Hongru Yuan, Chengpeng Li, Guanting Dong, Chang Lu, Yuanfeng Chen, Jiwen Zhou, Chuanqi Wang, Changyu Wu, Pengfei Zeng, et al. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[17] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.