Reinforcement Learning for Long-Horizon Multi-Turn Search Agents

Vivek Kalyan
Singapore
hello@vivekkalyan.com

Martin Andrews
Red Cat Labs, Singapore
martin@redcatlabs.com

Abstract

Large Language Model (LLM) agents can leverage multiple turns and tools to solve complex tasks, with prompt-based approaches achieving strong performance. This work demonstrates that Reinforcement Learning (RL) can push capabilities significantly further by learning from experience. Through experiments on a legal document search benchmark, we show that our RL-trained 14 Billion parameter model outperforms frontier class models (85% vs 78% accuracy). In addition, we explore turn-restricted regimes, during training and at test-time, that show these agents achieve better results if allowed to operate over longer multi-turn horizons.

1 Introduction

Recent advances in LLM agents (Wang et al. [1], Li [2, 3]) have shown impressive capabilities in tool use (Qu et al. [4]) and multi-step reasoning (Wang et al. [5]). This has led to a growing interest in their application to complex, long-horizon interactive tasks such as multi-turn document search, where an agent must interact with a document collection over several turns to locate specific information.

Reinforcement Learning (RL, Wen et al. [6]) offers a promising framework for training agents in these interactive settings. The successful retrieval of a document provides a natural, verifiable reward signal that can be used to optimise the agent's behaviour programmatically. In this work, we explore the application of RL to multi-turn search agents in the legal domain.

Our key contributions are:

- Showing that on a legal dataset, a 14B RL-trained model is able to outperform frontier models accessible only through APIs; and
- Exploring how RL-trained models can take advantage of the multi-turn setting by running experiments in which the number of turns is restricted - both during training and at test-time.

2 Related Work

Retrieval Augmented Generation (RAG, Lewis et al. [7], Gao et al. [8]) has emerged as a promising solution for incorporating knowledge from external databases in the LLM era, helping to combat the challenges of hallucinations, outdated knowledge, and non-transparent, untraceable reasoning processes. However, many implementations are either based around a single-retrieval phase Lewis et al. [9], or a pre-set process to determine the retrievals (Jiang et al. [10]).

Reinforcement Learning libraries, such as Agent Reinforcement Trainer (ART, Hilton et al. [11]), simplify the RL training of tool use by LLM agents, allowing for techniques such as Chain of Retrieval (Wang et al. [12]) to be readily implemented. In this work, we use ART to investigate the role of multi-turn behaviour for RAG tool use on a legal search benchmark.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 Workshop on Multi-Turn Interactions in Large Language Models.

3 Methods

3.1 Task and dataset

We construct a legal search benchmark from 5 years of Singapore court judgments. Each document in the dataset was parsed into an XML format, preserving its structural hierarchy with unique IDs for each section (e.g., `2021_SGCA_3:judgement:introduction:p1`).

Synthetic question-answer pairs were generated through a multi-stage pipeline: (i) extracting patterns from seed queries provided by practicing lawyers; (ii) generating 10 candidate question/answer pairs per document using Gemini 2.5 Pro (Gemini Team [13]); and (iii) aggressively filtering these candidates based on criteria of realism, difficulty, and variety. The final dataset contains 2,300 questions with ground-truth documents, questions and answers.

3.2 Naïve RAG baseline

We established a single-turn RAG baseline where, given a query, the system: (i) Executes searches using (a) BM25 (Manning et al. [14]) keyword search and (b) FAISS (Douze et al. [15]) semantic search using the raw query; (ii) Combines results from both methods; and (iii) Prompts the LLM to answer based solely on the retrieved context.

This baseline approach mirrors production RAG systems optimized for latency, making a single retrieval attempt without refinement or follow-up searches. Since a model cannot request additional information or explore related sections, this naïve RAG baseline forces it to work with whatever the initial retrieval returns.

3.3 Models used in RL training

The base model used for the RL training experiments was Qwen3-14B (Yang et al. [16]), whereas the Qwen3-0.6B model served as a low-cost alternative during code development, enabling the use of a much smaller GPU set-up. To save on resources, only LoRA adapter (Hu et al. [17]) components were trained. For the Reward Model, we used Gemini 2.5 Pro (Gemini Team [13]), which produced satisfactory binary decisions about the quality of the roll-out responses.

3.4 Agent architecture

All of the agents used in this work had access to three complementary tools for document exploration:

- **Keyword search**: Takes a query string and returns K results from BM25 retrieval over document paragraphs. Each result contains the section ID and a snippet highlighting the matched terms.
- **Semantic search**: Takes a natural language query and returns K results using cosine similarity over FAISS-indexed `all-MiniLM-L6-v2` (Song et al. [18]) embeddings. The returned results included section IDs and relevant snippets, enabling conceptual rather than lexical matching.
- **Read document content**: Takes a section ID and returns the complete content of that section. The hierarchical ID structure (e.g. `A:B:C`) also enables navigation: Agents can "hop" to parent sections by truncating IDs (`A:B:C` $\rightarrow$ `A:B`)

This system design creates a two-phase search pattern: broad exploration via keyword/semantic search to identify promising documents, followed by targeted reading to extract specific information.

The agent loop starts with the system prompt and query, and parses out {`<think>`, `<tool>`, and `<answer>`} sections from the response. Tool calls are executed and the results are returned to the model, continuing until the model produces an answer.

For details of the system prompt used for the Agentic settings, please see Appendix A.1.

3.5 Reinforcement Learning

For the RL training, we used the ART library, which in turn used Parameter-Efficient Fine-Tuning (PEFT, Mangrulkar et al. [19]); `unsloth` from Daniel Han and team [20]; and the Transformer Reinforcement Learning library (`trl`, von Werra et al. [21]) to train model LoRA adapters.

Table 1: Performance comparison of multi-turn agents on legal document search

Model	Accuracy (%)	Avg. Turns
Naïve RAG (Gemini 2.5 Pro)	33	1.0
Qwen3-14B (base)	53	3.7
Gemini 2.5 Flash	66	3.4
Gemini 2.5 Pro	78	5.3
OpenAI o3	81	7.1
Qwen3-14B + RL	**85**	6.2

For each query, vLLM (Kwon et al. [22]) was used to generate multiple trajectories from the model, from which RL rewards (detailed below) were calculated for each roll-out. YaRN (Peng et al. [23]) was used to extend the context of vLLM to 128k tokens to allow for extended multi-turn roll-outs. Following the reward evaluation, Group Relative Policy Optimization (GRPO, Shao et al. [24]) was used to optimise the model policy. At the end of each step, the LoRA adapters used by vLLM were updated, so that the following roll-outs used the updated policy. During training, `group_size` was set to 6, and 8 groups were run per step.

A system of partial rewards was found necessary to enable Reinforcement Learning to work effectively. Our reward structure created distinct behavioural bands that guided the model toward desired outcomes by ensuring that even failed trajectories provided learning signal:

- **[1.0, 2.0]**: Correct answer with proper citations. Higher rewards for fewer turns/searches
- **[0.0, 1.0]**: Model returns "I don't know" when unable to find sufficient evidence (preferable to hallucination)
- **[-1.0, 0.0]**: Incorrect answer provided. Partial credit (+0.1 each) for finding correct documents
- **[-2.0, -1.0]**: Formatting errors preventing tool execution (malformed tool calls, invalid arguments, non-existent document IDs)

Progress rewards (finding the right document, reading it, correct source citation) help the model learn intermediate skills necessary for the full task. Efficiency bonuses encourage models to achieve correct answers with fewer searches and turns. Critically, the above reward structure penalizes hallucination more severely than admitting uncertainty, training the model to say "I don't know" rather than fabricate answers when evidence is insufficient.

See Appendix A.2 for more information about the metrics used for tracking agents during RL-training.

3.6 Turn-restricted evaluation

To understand how models utilize multiple turns, we force early termination of the agent multi-turn tool-use by prefixing `<answer>` to the assistant message at turn N, forcing the model to produce an answer. Concretely, an N-turn roll-out has the following flow :

$$\texttt{query} \rightarrow \texttt{response} \rightarrow \{\texttt{reformulate search} \rightarrow \texttt{response}\}^{\wedge}N \rightarrow \texttt{answer}$$

Thus, a 0-turn rollout corresponds to naïve RAG.

4 Results

4.1 Overall performance

Table 1 shows that the RL-trained Qwen3-14B achieves 85% accuracy, surpassing all other models listed including frontier models that are only accessible via API. The progression from naive RAG (33%) to multi-turn interaction shows clear benefits of iterative search, with each tier of model capability yielding substantial improvements.

The Qwen3-14B model without RL training plateaus at 53% accuracy despite having access to the same tools and multi-turn interactions, highlighting that tool access alone is insufficient without learning how to use the capability effectively.

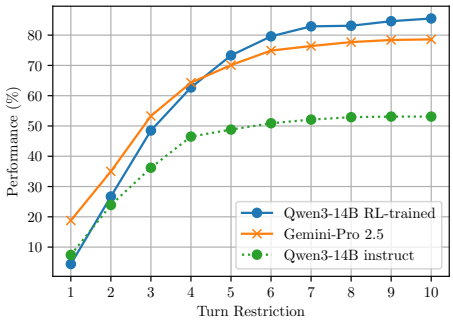

Figure 1: Performance of multi-turn agents under turn restrictions

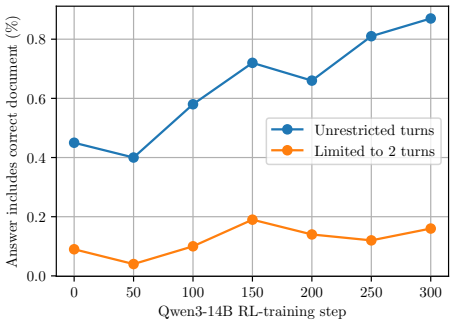

Figure 2: Effect of restricting turns during RL training

4.2 Impact of turn-restricted inference

To quantify the relationship between multi-turn interaction and model performance, we evaluated selected models under turn restrictions using the methodology described in Section 3.6.

Figure 1 shows the performances for each model under 1 to 10 turn restrictions. All models exhibit monotonic improvement with additional turns, confirming that iterative search is essential for this task. The more significant finding is that the models diverge in their ability to exploit additional search opportunities : The base Qwen3-14B performance plateaus after 6 turns, while the RL-trained variant and Gemini 2.5 Pro continue improving throughout all 10 turns.

In addition, Gemini Pro 2.5 outperforms the RL-trained model in the low (less than 5) turn regime, whereas the far smaller RL-trained model shows additional benefit of multi-turn interactions beyond that. This phenomenon demonstrates a limitation of prompt-based approaches: while they can execute multi-turn search, they lack learned exploration strategies that compound value across multi-turn interactions. This suggests that RL might play an important role in creating strong search agents.

4.3 Impact of turn-restricted training

To investigate whether effective multi-turn behaviour can be learned in turn-constrained settings, we also trained a Qwen3-14B model with a limit of 2 turns imposed during training, otherwise following the methodology of Section 3.6, with the plan of testing its generalisation capability later.

Figure 2 tracks the percentage of trajectories containing correct document citations across training steps for both the unrestricted and 2-turn restricted models. While the unrestricted model shows steady improvement from approximately 40% to 85% correct document identification over 300 training steps, the 2-turn restricted model exhibits no meaningful learning progress, fluctuating around its initial baseline of 10-15% throughout training.

This failure to improve stems from the fundamental requirements of GRPO, which learns by comparing relative rewards within trajectory batches. With only 2 turns available, the model is unable to achieve correct answers at a high enough rate, providing insufficient positive examples for learning.

5 Discussion

Reinforcement learning enables agents to learn effective tool use through practice rather than instruction. When agents encounter new tools or unfamiliar document collections, RL allows them to develop expertise through trial and error, discovering which search strategies work best. This approach is applicable to other tasks with multi-turn interactions, where agents must decide how to use their turns wisely, and this is an area of active, ongoing research.

Acknowledgments and Disclosure of Funding

This work's reinforcement learning experiments were significantly simplified by the use of the open-source ART library from OpenPipe. We acknowledge its crucial role in abstracting away low-level infrastructure tasks, including rollout management and weight updates.

Support for this research was provided by the Google AI Developer Programs team, including access to the Gemini models and GPUs on Google Cloud Platform.

The authors would also like to thank the reviewers for the NeurIPS 2025 Workshop on Multi-Turn Interactions in LLMs for their time and valuable feedback.

References

[1] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024. ISSN 2095-2236. URL http://dx.doi.org/10.1007/s11704-024-40231-1.

[2] Xinzhe Li. A review of prominent paradigms for LLM-based agents: Tool use, planning (including RAG), and feedback learning. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9760–9779, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.652/.

[3] Xinzhe Li. A survey on LLM test-time compute via search: Tasks, LLM profiling, search algorithms, and relevant frameworks. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=x9VQFjtOPS.

[4] Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with LLMs: A survey. *Front. Comput. Sci.*, 19(8), August 2025.

[5] Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. Offline reinforcement learning for LLM multi-step reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8881–8893, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.464. URL https://aclanthology.org/2025.findings-acl.464/.

[6] Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs, 2025. URL https://arxiv.org/abs/2506.14245.

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for Large Language Models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[10] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, 2023.

[11] Brad Hilton, Kyle Corbitt, David Corbitt, Saumya Gandhi, Angky William, Bohdan Kovalenskyi, and Andie Jones. ART: Agent reinforcement trainer. https://github.com/openpipe/art, 2025.

[12] Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation, 2025.

[13] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

[14] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library, 2024.

[16] 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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[18] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: masked and permuted pre-training for language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[19] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

[20] Michael Han Daniel Han and Unsloth team. Unsloth, 2023. URL http://github.com/unslothai/unsloth.

[21] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[23] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of Large Language Models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

[24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

A Appendix

A.1 Prompting

The System Prompt for non-thinking models was as follows:

```
You are a legal research assistant that can search legal documents to answer
    questions.

You have access to the following tools:

- search_keyword(query: str, num: int) -> str: Search using keyword/BM25 search for
    exact term matches.
- search_semantic(query: str, num: int) -> str: Search using semantic/vector search
    for conceptual similarity.
- read_document_part(part_id: str) -> str: Read a document part by ID. Part IDs use
    hierarchical format (e.g., A:B:C). To access parent parts, remove the last
    segment (e.g. A:B:C -> parent is A:B).

You may call one tool per turn, for up to {max_turns} turns, before giving your
    final answer.

In each turn, you should analyze what information you need and respond with EITHER a
     tool call OR your final answer.

For tool calls, use this format:
<think>
[your reasoning for what to search for and why]
</think>
<tool>
{"name": "tool_name", "args": {"query": "search query"}}
</tool>

When you have enough information, give your final answer in this format:

<think>
[your reasoning for the answer]
</think>
<answer>
[your comprehensive answer citing the evidence you found or "I don't know" if you
    didn't get enough information]

<sources>
<source>doc_id_1</source>
</sources>
</answer>
```

For the 'Thinking models', such as o-3 and Gemini Pro, we simply omit the instructions about the usage of `<think>` tags, and the thinking budgets were set to default values.

A.2 Metrics

We tracked 13 distinct metrics across four categories to comprehensively assess agent performance:

Final Outcome Metrics:

- `answer_correct`: Whether the answer matches ground truth (evaluated via LLM judge)
- `sources_correct`: Whether cited documents match ground truth documents (verifiable)
- `returned_i_dont_know`: Whether the model explicitly states uncertainty (verifiable)
- `attempted_answer`: Whether the model provided any answer (verifiable)

Progress Tracking:

- `ever_found_right_doc`: Whether correct document appeared in any search results (verifiable)
- `ever_read_right_doc`: Whether the model used the read tool on the correct document (verifiable)

Formatting Errors:

- `cant_parse_tool_call`: Malformed JSON or missing required tags (verifiable)
- `bad_tool_call_name`: Invalid tool name specified (verifiable)
- `bad_tool_call_args`: Incorrect arguments for valid tool (verifiable)
- `bad_sources_id`: Referenced non-existent document IDs (verifiable)

Efficiency Metrics:

- `num_turns`: Total number of tool-use turns taken (verifiable)
- `num_searches`: Count of keyword/semantic searches executed (verifiable)
- `ran_out_of_turns`: Whether the turn limit was reached (verifiable)

Notably, 12 of 13 metrics are verifiable without requiring an external judge, enabling fast and deterministic evaluation during training. Only `answer_correct` requires LLM evaluation, for which we use Gemini 2.5 Pro to provide a True/False binary classification.