RESEARCH-ARTICLE

Log Pruning for Efficient Q&A in 5G Networks

ALI MAMAGHANI, University of California, San Diego, San Diego, CA, United States

QINGYUAN ZHENG, University of California, San Diego, San Diego, CA, United States

USHASI GHOSH, University of California, San Diego, San Diego, CA, United States

VICRAM RAJAGOPALAN, Texas A&M University, College Station, TX, United States

SRINIVAS SHAKKOTTAI, Texas A&M University, College Station, TX, United States

DINESH BHARADIA, University of California, San Diego, San Diego, CA, United States

Open Access Support provided by:

University of California, San Diego

Texas A&M University

Published: 04 November 2025

Citation in BibTeX format

ACM MobiCom '25: The 31st Annual International Conference on Mobile Computing and Networking
November 4 - 8, 2025
Hong Kong, China

Conference Sponsors:
SIGMOBILE

Log Pruning for Efficient Q&A in 5G Networks

Ali Mamaghani[1], Qingyuan Zheng[1], Ushasi Ghosh[1], Vicram Rajagopalan[2], Srinivas Shakkottai[2], and Dinesh Bharadia[1]
[1]University of California San Diego, CA, USA, [2]Texas A&M University, TX, USA

Abstract

Large Language Models (LLMs) excel across many tasks but struggle in specialized domains like system logs, often leading to inaccuracies. Retrieval-Augmented Generation (RAG) mitigates this by grounding responses in external knowledge, yet conventional RAG faces scalability issues with massive structured datasets due to the high computational cost of embeddings and vector operations. We introduce a novel RAG architecture tailored for large-scale structured data. By combining intelligent context pruning with signal processing techniques that exploit inherent log structures, our design enables rapid, efficient retrieval with reduced computational overhead while maintaining accuracy. We validate this approach on a real-world, high-throughput 5G codebase generating millions of log entries per minute, alongside a comprehensive Q&A dataset on these logs. Results show substantial gains in both speed and accuracy over traditional RAG, making this solution well-suited for real-time analytics in large-scale operational environments.

CCS Concepts

• **Networks → Network measurement**; • **Software and its engineering → Software creation and management**; • **Information systems → Data management systems**.

Keywords

5G, Log Processing, LLM, RAG, Pruning

ACM Reference Format:

Ali Mamaghani[1], Qingyuan Zheng[1], Ushasi Ghosh[1], Vicram Rajagopalan[2], Srinivas Shakkottai[2], and Dinesh Bharadia[1]. 2025. Log Pruning for Efficient Q&A in 5G Networks. In *Open & AI RAN 2025, November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3737900.3770171

1 Introduction

The advent of 5G networks ushers in unprecedented scale, dynamic configurations, virtualization, and multi-access edge computing, generating vast real-time streams of operational log data. With massive IoT, eMBB, and URLLC driving exponential growth, these logs—often reaching gigabytes per minute per network slice or edge site—are critical for ensuring stringent SLAs and enabling root cause analysis in highly distributed environments. Yet, manually navigating such heterogeneous datasets is infeasible, and conventional automated pipelines for parsing, compression, and anomaly detection often fail to provide deep, contextual insights into real-time system state and behavior.

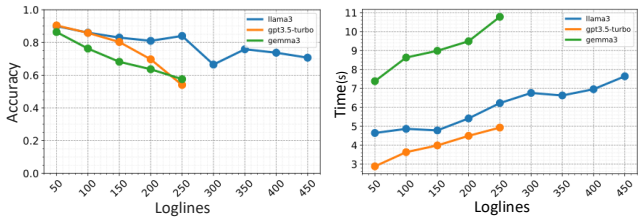
Crucially, network logs implicitly capture the system's operational state machine over time. The ability to issue natural language question-and-answer (Q&A) queries directly on these streams represents a paradigm shift, enabling timely, granular insights into performance, faults, and state transitions that are otherwise buried in fragmented architectures.

Advances in Large Language Models (LLMs), with their exceptional natural language understanding, now make it feasible to build automated Q&A agents that interpret complex log patterns and bridge human intent with the vast landscape of 5G system logs. While LLMs hold great promise for log analysis, directly feeding raw logs is impractical due to limited context windows (thousands of tokens vs. gigabytes per minute in 5G). Retrieval-Augmented Generation (RAG) addresses this by selecting only relevant fragments, but applying it to massive, continuous log streams remains slow and compute-intensive. As shown in Figure 1, expanding input context not only degrades accuracy but also increases response latency, underscoring the need for effective context pruning. Prior efforts in pruning, parsing, and compression help reduce input size, yet fall short at 5G scale. Compression, for instance, removes redundancy but without semantic filtering still passes irrelevant or noisy entries from heterogeneous 5G elements. Consequently, existing methods mitigate some challenges but fail to deliver the efficiency and generalizability needed for real-time, large-scale log reasoning in dynamic 5G networks

We address the critical challenge of enabling real-time, interactive question answering over massive, high-throughput 5G log streams. We propose a novel real-time Retrieval-Augmented Generation (RAG) pipeline designed to overcome the latency bottlenecks of conventional RAG. Our key contribution is a high-speed context retrieval mechanism that

OpenRan '25, November 4-8, 2025, Hong Kong, China

ACM ISBN 979-8-4007-1977-6/25/11
https://doi.org/10.1145/3737900.3770171



(a) Accuracy vs number of log lines (b) Generation time vs number of log lines

Figure 1: System accuracy and generation time

operates directly on incoming logs. Unlike traditional approaches that index large historical corpora, our method processes logs as they arrive (e.g., mapping strings to numeric time-series) and rapidly correlates them with query-derived patterns. This correlation bypasses slow index lookups, acting as an out-of-the-loop accelerator that quickly extracts relevant segments for the LLM, enabling genuine real-time interrogation at scale.

Log data, however, introduces unique challenges. Entries are dynamic and time-sensitive, dominated by timestamps, numerical values, and domain-specific abbreviations—features with low semantic richness that hinder direct interpretation. LLMs must perform semantic inference on such inputs, but limited context windows and fragmented data increase hallucinations and reasoning errors. Moreover, conventional text-based RAG pipelines are poorly suited for evolving log streams where rapid, low-latency synthesis is essential. Without efficient retrieval and context generation, RAG cannot meet the demands of 5G monitoring, fault diagnosis, and anomaly detection.

To address these challenges, we develop a log-aware RAG retrieval pipeline for real-time 5G log analysis, and implement it on an O-RAN–compliant testbed that enables controlled scenario creation under both split 8 and split 7.2 architectures. Building on this testbed, we further construct a large-scale 5G log Q&A dataset spanning diverse operational scenarios, which we use to rigorously evaluate and benchmark our approach.

2 Related Work

Retrieval-Augmented Generation (RAG). RAG has emerged as a powerful paradigm for enhancing the performance of LLMs by incorporating external knowledge without modifying the model parameters [4, 8, 11]. In a typical RAG pipeline, a dense retriever first selects relevant top-k contexts from an external corpus [7, 10], which are then fed into an LLM to generate accurate and grounded responses. This mechanism enables LLMs to handle long-tail knowledge and provide up-to-date information [17]. However, the limited context windows of large language models require sophisticated context pruning or selection mechanisms to identify the most relevant information from potentially vast retrieval results.

Context Pruning. Context pruning approaches improve RAG scalability by reducing the sizes of contexts to be used for generation. Many techniques identify and remove the least relevant portions of the context [1, 5, 9, 12, 13, 16]. Related techniques summarize or rephrase the context to reduce its size [14, 15]. Many context selection techniques exist, but applying these to the unique structure and temporal nature of log data is not straightforward. As seen in a comparison of various state-of-the-art context selection methods [1], most methods achieve a compression ratio of 4x or less, which is insufficient given the the massive scale of system log data.

Log-Based Question-Answering. Specifically for the log domain, LogQA [3] introduces a framework for log-based question answering where a retriever model is trained using supervised learning to select relevant log snippets given a query. Although effective within a specific domain, this approach suffers from a key limitation: it requires system-specific training data. This means a new retriever model must be trained for each type of system log, limiting its generality and requiring significant effort to adapt to new environments or log formats.

3 Testbed and Dataset

This section details the experimental setup of a 5G Standalone network testbed built with open-source software and commercial hardware. It explains how this testbed is configured to collect highly granular, DEBUG-level logs from every layer of the gNB protocol stack. Finally, it outlines the process of using GPT-3.5 and human validation to convert these raw log files into a clean, high-quality question-and-answer dataset for evaluation.

3.1 Experimental setup

Our experimental setup is a full end-to-end 5G standalone (SA) testbed with two complementary architectures (Figure 2). On the left, we deploy a split-8 configuration: the 5G Core (5GC) and gNB run on an Amarisoft UE emulator connected to a USRP X410 SDR as the Radio Unit (RU), establishing over-the-air links to both emulated and COTS UEs (Quectel 5G modem, OnePlus 8T, Pixel 8). On the right, we implement a split-7.2 architecture with CU, DU, and 5GC built from the srsRAN codebase on a high-performance server. The split-7.2 RU (Foxconn RPQN7801E) is synchronized via a PTP grandmaster clock and integrated with NVIDIA ARC-OTA edge server. In both setups, DEBUG-level logging and packet captures are enabled across the gNB stack, enabling fine-grained cross-layer analysis with heterogeneous UEs.

3.2 Log Generation

During log generation, we developed automated scripts to manage scenario configuration and execution in a unified

Case	Scenario	Log Level	Runtime	Log Size
1	1UE: UE Disconnection and New UE Registered	Global INFO; DEBUG Across PDCP/RLC/MAC/F1	120s	0.96 GB
2	1UE: UE SNR Drop under Adverse Channel Conditions		90s	0.93 GB
3	1UE: Mobility (COTS UE in Fast Motion)		90s	1.07 GB
4	1UE: Congested Network (Load Exceeds Capacity)		120s	1.91 GB
5	2UE: Higher DL than UL (both UEs)	Global DEBUG	90s	1.61 GB
6	2UE: UE1 UL, UE2 DL (UE2 Higher Rate)		90s	2.76 GB
7	4UE: Three Normal, One Very High Rate		90s	1.36 GB
8	4UE: UE1/UE2 Higher UL; UE3/UE4 Higher DL		90s	1.41 GB

Table 1: Scenario types, modes, and corresponding log sizes

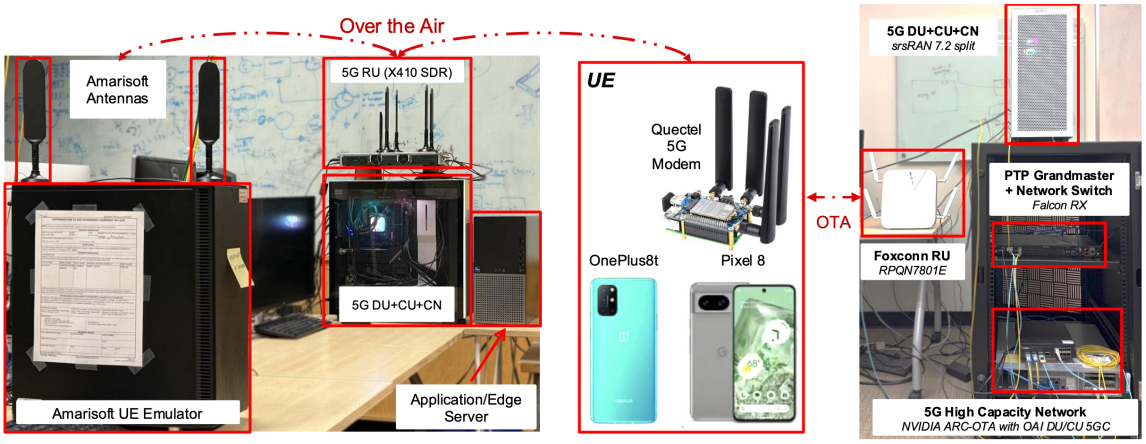


Figure 2: Experimental 5G Standalone (SA) network testbed.

manner. Each script automatically records the execution timestamp of every command. For different scenarios (shown in Table 1), traffic was generated using `iperf` under both ZMQ and Over-the-Air (OTA) modes, covering both TCP and UDP transmissions. By configuring multi-user cases (2UE/4UE) with symmetric or asymmetric uplink/downlink rates, varying channel conditions (e.g., SNR degradation, mobility), and network congestion, we collected logs ranging from 0.9 GB to 2.8 GB. As shown in Table 1, even a short 2-minute run generates logs at the gigabyte scale. Without pruning, such massive data would easily overwhelm the context window of LLMs, leading to inefficient or even inaccurate question answering. The scale and diversity of these experimental logs ensure a broad coverage of representative 5G operating conditions, while the precise execution timestamps provide reliable ground truth to validate answers in log-based Q&A tasks.

3.3 Dataset generation

We created a high-quality instruction-based dataset from raw log files using a combination of AI generation and human oversight. Each log line was provided as a prompt to a large language model (e.g., GPT-3.5), which generated a relevant question–answer pair. This automated process produced a large volume of Q&A samples, which were then carefully reviewed by human validators, with incorrect or irrelevant pairs discarded. The result is a clean, human-verified dataset that accurately reflects the source logs and is well-suited for evaluating our design.

4 System Design

The system architecture is shown in Fig. 1, which mainly consists of three modules: pruning, inference and reasoning.

The pipeline consists of three key stages: preprocessing, pruning, and inference.

4.1 Preliminaries

Log Preprocessing. The preprocessing phase standardizes heterogeneous log data into a uniform, query-friendly format. Given the semi-structured nature of logs, raw data originating from diverse sources, including flat key–value records and deeply nested JSON, poses significant challenges for log vectorization. To address this, we enforce a consistent structure by separating each log into two components: a

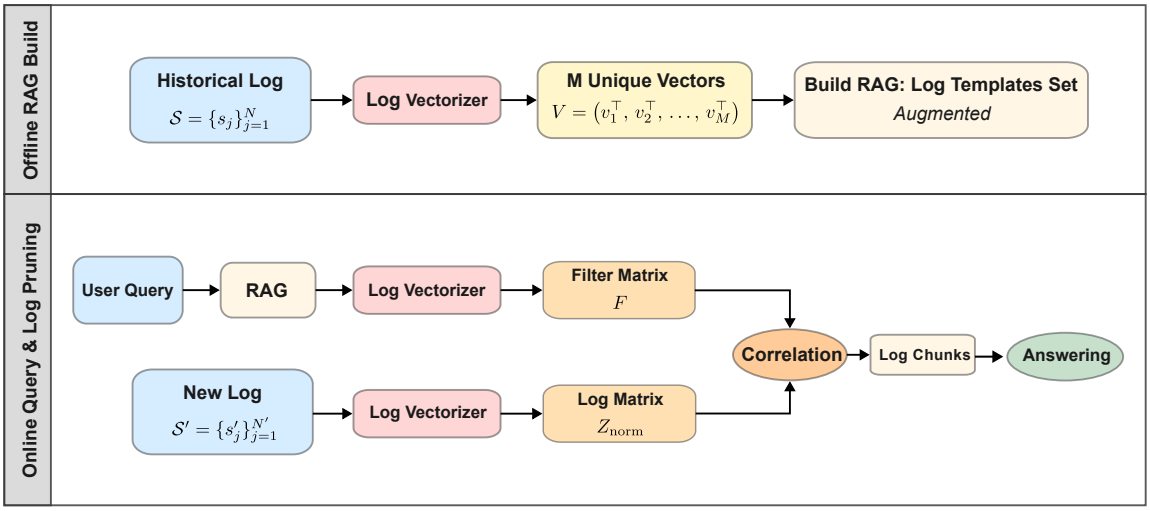


Figure 3: Log Pruning System Overview.

timestamp that serves as a temporal anchor and an information block that consolidates key diagnostic content. This normalization facilitates template detection, supports efficient indexing, and enables accurate semantic similarity matching.

Log Vectorization. During the offline stage, we build an n-gram vocabulary from the historical corpus $\mathcal{S}$. Each log is converted into sparse unigram–bigram features, forming a global vocabulary of size M. The vocabulary is embedded into a matrix $V \in \mathbb{R}^{M \times d}$ and normalized row-wise, yielding V_{norm}. This representation preserves recurring character patterns while filtering out dynamic fields such as timestamps and identifiers.

New logs $\mathcal{S}'$ are cleaned and mapped into vectors $Z_{\text{norm}} \in \mathbb{R}^{N' \times M}$. Cosine similarity between Z_{norm} and V_{norm} produces a score matrix S, from which each log s'_j is assigned to its nearest template:

$$\text{idx}_j = \arg\max_{1 \le i \le M} S_{j,i}.$$

The resulting ID sequence idx transforms the log-stream into template-level representations, enabling efficient indexing and retrieval.

RAG Augmentation. We use extracted log templates as the RAG basis and enrich them with domain knowledge to improve retrieval quality and semantic alignment. Specifically, we augment each log template with explanatory context derived from protocol specifications and implementation details. Linking each template to relevant context improves interpretability, match accuracy, and disambiguates abbreviations in logs. For example, in following log template:

```
[RLC ] [D] du=<:NUM:> ue=<:NUM:> <:*:> UL: RX
SDU. payload_len=<:NUM:> dc=data p=<:NUM:> si=full
sn=<:NUM:>
```

"si" denotes the segmentation-offset state in the 5G RLC layer, which generic models cannot infer without domain grounding. Augmentation ensures that such terms are correctly interpreted during retrieval. Finally, we encode the augmented templates using the "BAAI/bge-large-en-v1.5" sentence embedding model and indexes them with FAISS [2, 6].

4.2 Offline RAG Build

The offline RAG build stage establishes the foundation for efficient retrieval by transforming heterogeneous historical logs into a compact template set. As shown in Figure 3, the process starts with the historical corpus S, which is normalized to ensure a uniform structure across diverse formats.

Each normalized log is processed by the *Log Vectorizer*, which extracts character-level n-gram features and masks dynamic fields (e.g., timestamps, identifiers), producing sparse structural representations that capture recurring patterns while filtering incidental variability.

The resulting vectors are aggregated and deduplicated into M unique templates, denoted V. This template library serves as the semantic dictionary for the RAG system and is further expanded with LLM-based augmentation to improve alignment with user queries.

4.3 Online Query and Log Pruning

At runtime, we have two parallel streams, one is RAG Retrieval, well another stream is new log processing.

Pruning Method	Model	Accuracy (%)	Compression (%)	Generation Time (s)
No Pruning	GPT-3.5	54.0	0.0	2.84
	GPT-4o-mini	91.0	0.0	7.57
	LLAMA-3	70.7	0.0	7.64
	GEMMA-3	57.5	0.0	10.78
8-Template Pruning	GPT-3.5	77.2	82.5	0.91
	GPT-4o-mini	76.3	83.8	2.57
	LLAMA-3	71.2	82.0	5.37
	GEMMA-3	70.2	82.5	8.74
32-Template Pruning	GPT-3.5	85.3	60.2	0.96
	GPT-4o-mini	89.3	62.9	3.32
	LLAMA-3	85.9	60.3	6.09
	GEMMA-3	73.7	60.2	9.44

Table 2: Comparison of pruning methods and their impact on accuracy, compression, and generation time

RAG Retrieval. In this stage, the user query is compared against the augmented template set $\widetilde{T}$. The top-k most similar templates are selected, with their indices denoted by $\mathcal{I}$, and the corresponding entries from the original set T are retrieved. Each selected template is then encoded into a unigram–bigram vector representation, and the resulting collection is stacked into the *Filter Matrix* $F \in \mathbb{R}^{k\times d}$. Acting as a high-precision gatekeeper, F filters out irrelevant log structures and forwards only the most semantically aligned templates to downstream classification. The functionality of this module is illustrated in Example 1.

Example 1: RLC Retransmission Query. For a query "Why frequent uplink retx?", top-k templates often include patterns like `[RLC] PDU, ..., RETX, ... , sn=. . .` and `STATUS report update`. Their rows in F will emphasize bigrams tied to `RETX`, `STATUS`, and `sn`, steering subsequent pruning toward those structures.

New Log Processing. In parallel, we process the stream of new logs, filtering them with respect to timestamp to ensure temporal relevance to the query. These logs are then vectorized to form the Log Matrix, where each vector encodes a single log entry. We describe the vectorization process in the *Log Vectorization* section.

Log Selection. To perform local log selection, we first determine the indices of the expected templates within the new log chunks and then compute the corresponding relevance scores:

$$M \;=\; Z_{\text{norm}}\, F^{\top} \;\in\; \mathbb{R}^{N'\times k}$$

Then, using the following expression, we identify the rows that are highly consistent with our extracted templates:

$$Index \;=\; \Big\{\, i \;\Big|\; \max_{1\le j\le k} M_{i,j} > Threshold \Big\}.$$

Using the new log lines as $\mathcal{S}'$ in the **Log Vectorization** section, we can have the final retrieved lines as: $\text{MatchedLines} = \{\, s_i \mid i \in Index\}$. We demonstrate the operation of log selection using Example 2.

Example 2: Timestamps + Query. For a query scoped to $[t_0, t_1]$, only logs in this window are vectorized. If F is dominated by RLC-reassembly patterns, then $\max_j M_{i,j}$ spikes precisely on lines containing `RX SDU`, `sn`, `si`, etc., while EL1/PHY noise is pruned out. This yields a compact, query-aligned context for question answering.

5 Experiments

To rigorously evaluate the effectiveness of our system, we conducted a series of experiments for a variety of LLMs. We performed a quantitative assessment of our pruning strategy, examining its compression ability, generation time, and accuracy in direct question-and-answer tasks.

5.1 Benchmark Results

The pruning module is a core innovation of our system, designed to optimize downstream retrieval and inference by judiciously selecting only the most relevant log templates and their associated entries. To demonstrate the efficacy of this approach, we performed an experiment comparing the accuracy achieved by our system—which feeds a precisely pruned log context, even approaching the maximum token size—to the accuracy obtained when directly presenting raw, unpruned logs to an LLM. We constructed a dataset containing large log entries, with realistic parameter distributions and timestamp metadata. A set of 198 queries were generated, each associated with a ground truth mapping to the correct log templates required for accurate answers. For each query, we measured:

Answer Accuracy is the proportion of questions where the LLM's output correctly matches the expected answer. To evaluate this, we use a sophisticated LLM, such as **DeepSeek-R1**, as a judge. This judge model is given the question and the generated answer, then evaluates its correctness based on a predefined rubric. This method is highly scalable and provides more nuanced feedback than traditional methods, as the judge can explain its reasoning for the score it gives.

Compression is the ratio of the pruned tokens from the input of LLM. This metric measures the efficiency of pruning in the input data that the model needs to process to generate an answer. A higher compression can decrease both computational cost and processing time.

Generation Time is the total time it takes for the LLM to produce a complete answer after receiving the input. This includes the time spent processing the input and generating the output tokens. It is a critical measure of the system's latency and overall speed.

Table 2 demonstrates that pruning can be an effective technique to improve both accuracy and generation time for language models. Pruning reduces the number of tokens an LLM must process, which leads to a faster response. For models like GPT3.5 and LLAMA-3, pruning also significantly boosts accuracy, suggesting that it helps the models focus on the most relevant information without getting distracted by noise in the original input. However, the effect of pruning can vary by model. For a highly capable model like GPT-4o-mini, an aggressive pruning method (e.g., 8 templates) may harm performance. In contrast, a more moderate approach (32 templates) can restore its accuracy while still cutting costs. The trade-off between the degree of compression and accuracy is evident, with 32 templates offering a more balanced outcome for most models compared to the 8-template approach.

5.2 Q&A examples

The following examples illustrate how our pruning methodology improves accuracy by removing irrelevant content. Expanding the template set further enhances retrieval, ensuring logs relevant to the query are captured. We define the corresponding log entry as the specific entry containing the answer, selected from a large and diverse log set. Example 1 highlights how pruning preserves factual integrity, Example 2 shows how larger templates improve coverage, and Example 3 demonstrates how pruning enables accurate responses to high-level domain-specific questions.

As future work, we plan to expand to larger and more diverse log datasets, supporting more complex queries. This will enable reasoning-based tasks beyond simple retrieval, such as multi-step analyses across heterogeneous log events. Such capabilities are especially important in 5G, where performance issues often stem from cross-layer interactions. Advancing in this direction will support more comprehensive diagnostics, including anomaly detection, stability assessment, and root cause analysis.

Eg. 1: Answer with no/32 pruning (GPT3.5-turbo)

Corresponding log entry: 2025-05-11T20:41:57.747348 [RLC] [D] du=0 ue=2 DRB1 UL: Generating status PDU. rx_next=123544 rx_highest_status=123544 stop_sn=123544
Question: In the log with timestamp 2025-05-11T20:41:57.747348, what is the value of "rx_next" for DRB1 UL?
Ground Truth Answer: 123544
Generated answer with no template pruning: Based on the provided log, in the log with timestamp 2025-05-11T20:41:57.747348, the value of "rx_next" for DRB1 UL is **123547**.
Generated answer with 32-template pruning: Based on the provided log, the value of "rx_next" for DRB1 UL at timestamp 2025-05-11T20:41:57.747348 is **123544**.

Eg. 2: Answer with 8/32 pruning (GPT3.5-turbo)

Corresponding log entry: 2025-05-11T20:41:57.747345 [RLC] [I] du=0 ue=2 DRB1 UL: RX PDU. pdu_len=669 dc=data p=0 si=first sn=123544
Question: At timestamp 2025-05-11T20:41:57.747345, what is the length of the received Protocol Data Unit (PDU) in UL for DRB1?
Ground Truth Answer: 669 bytes
Generated answer with 8-template pruning: Based on the provided log, at timestamp 2025-05-11T20:41:57.747345, the length of the received Protocol Data Unit (PDU) in UL for DRB1 is **1504**.
Generated answer with 32-template pruning: Based on the provided log at timestamp 2025-05-11T20:41:57.747345, the length of the received Protocol Data Unit (PDU) in UL for DRB1 is **669 bytes**.

Eg. 3: Answer with 32-template pruning (GPT3.5-turbo)

Question: I set the bitrate to 120 Mbps, but why does UE bitrate drop to 43 Mbps?
Answer: The drop in UE bitrate from 120 Mbps to 43 Mbps could be due to several factors observed in the log:
Poll Retransmit Timer: The log shows multiple instances of the poll retransmit timer being started (e.g., lines 1, 97, 100, 612). This indicates potential retransmissions, which can reduce effective throughput.
Grant Length Variability: The grant lengths for PDUs vary significantly (e.g., lines 2, 5, 9, 13), suggesting inconsistent resource allocation, which can impact throughput.
Queued SDUs: The number of queued SDUs remains high (e.g., lines 3, 20, 27), indicating potential congestion or bottleneck issues.
These factors collectively contribute to congested network.

References

[1] Natalia Chirkova, Timothee Formal, Vassilina Nikoulina, and Stéphane Clinchant. 2025. Provence: Efficient and Robust Context Pruning for Retrieval-Augmented Generation. arXiv:2501.16214 doi:10.48550/arXiv.2501.16214 arXiv preprint arXiv:2501.16214.

[2] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[3] Shouheng Huang, Yao Liu, Christopher Fung, Jun Qi, Hongxu Yang, and Zhi Luan. 2023. LogQA: Question Answering in Unstructured Logs. arXiv:2303.11715 doi:10.48550/arXiv.2303.11715

[4] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 874–880. https://aclanthology.org/2021.eacl-main.74

[5] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 13358–13376. doi:10.18653/v1/2023.emnlp-main.825

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 6769–6781. https://aclanthology.org/2020.emnlp-main.550

[8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and Sebastian Riedel. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[9] Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing Context to Enhance Inference Efficiency of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 6342–6353. doi:10.18653/v1/2023.emnlp-main.391

[10] Shuwen-Cheng Lin, Akari Asai, Mingwei Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xinying Chen. 2023. How to Train Your Dragon: Diverse Augmentation Towards Generalizable Dense Retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, 1693–1712. https://aclanthology.org/2023.findings-emnlp.109

[11] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 9802–9822. doi:10.18653/v1/2023.acl-long.546

[12] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data Distillation for Efficient and Faithful Task-Agnostic Prompt Compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 963–981. doi:10.18653/v1/2024.findings-acl.57

[13] Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023. Learning to Filter Context for Retrieval-Augmented Generation. arXiv:2311.08377 [cs.CL] https://arxiv.org/abs/2311.08377

[14] Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving Retrieval-Augmented LMs with Context Compression and Selective Augmentation. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=mlJLVigNHp

[15] Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. CompAct: Compressing Retrieved Documents Actively for Question Answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 21424–21439.

[16] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=ZS4m74kZpH

[17] Yujia Yu, Wei Ping, Zihan Liu, Bowen Wang, and Jun You. 2024. RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs. (2024). Manuscript in preparation.