

SUMMEHRY: Large Language Models for Generating Temporal Patient Vignettes

Stanford CS224N Custom

Eashan Monga
Department of Statistics
Stanford University
eashan.monga@stanford.edu

Asmita Sood
Department of Biomedical Data Science
Stanford University
asmitas@stanford.edu

Arlina Shen
Department of Biomedical Data Science
Stanford University
ahshen26@stanford.edu

Abstract

Electronic health records (EHRs) are longitudinal and temporally complex, making it difficult to extract coherent, decision-relevant representations of patient trajectories. While large language models (LLMs) perform well on single-note summarization, generating clinically useful summaries over many encounters remains challenging because records are long, sequential, and easy to distort under repeated compression. We study whether LLMs can generate longitudinal patient vignettes that preserve diagnostic signals, maintain chronological coherence, and retain value for downstream medical tasks, such as tumor boards. We propose a recursive state-update framework in which the summary at time $n - 1$ is incrementally updated with new structured and unstructured EHR data to produce a summary at time t . Our prompting strategy enforces decision-signal preservation, explicit time anchoring, and overwrite rules to reduce summary drift. To address the scarcity of gold-standard data, we construct a silver-labeled dataset with PDSQI-9-based [1] quality filtering and fine-tune open-weight models. We evaluate the resulting vignettes along three axes: clinical note quality, retention across updates, and downstream predictive utility. Our results show that progressive summaries can achieve strong factual quality while making information loss measurable and partially repairable, and can preserve useful signal for downstream cancer prediction. More broadly, this work frames clinical summarization as a temporal state-tracking problem rather than a one-shot compression task.

1 Key Information to include

- Mentor: Caroline Choi
- External Collaborators (if you have any): Dr. Jason Fries
- Sharing project: No
- Late Days (if you are using any): 3

Team member	Late days remaining
Arlina Shen	6
Asmita Sood	1
Eashan Monga	4

Summary	Value
Total pooled	11
Team size	3
Late days applied for the report	3

2 Introduction

Electronic health records (EHRs) contain rich longitudinal information, including diagnoses, imaging, and clinical narratives spread across many encounters. Because this information is fragmented over time and spans very large volumes of text, clinicians must reconstruct patient state from many partial documents [2, 3]. Large language models (LLMs) have shown promise for clinical summarization and can sometimes match or exceed clinician-written summaries on structured evaluations [4], but most prior work still treats summarization as a one-time compression problem. This does not reflect clinical reasoning, which is inherently iterative as new information arrives [5]. Longitudinal summarization is challenging because EHRs often exceed model context limits, repeated updates can cause information erosion, and summaries must remain temporally coherent as diagnoses, treatments, and findings evolve [6]. These challenges motivate treating summarization as a temporal state-update problem rather than a static one-pass task. We propose a progressive summarization framework for generating longitudinal patient vignettes from EHR data. Rather than summarizing the full record at once, the method treats the patient summary as an evolving clinical state over chronologically ordered EHR segments $\{C_0, C_1, \dots, C_T\}$:

$$S_0 = f(C_0), \quad S_t = f(S_{t-1}, C_t),$$

where S_t is the updated patient state after incorporating segment C_t . To improve stability, we add retention verification and repair. After each update, an auditing module compares the new summary with the prior state, identifies clinically relevant dropped facts, and reinserts critical omissions through a repair step. We evaluate this framework on longitudinal EHR records using three criteria: clinical quality, retention behavior, and downstream predictive utility.

3 Related Work

Clinical NLP has long studied how to extract structured information from EHRs for decision support and prediction, evolving from rule-based and statistical methods to deep learning approaches over clinical text and multimodal EHR data [2, 3]. Transformer-based models such as ClinicalBERT and BioBERT have achieved strong performance on tasks like entity recognition, relation extraction, and outcome prediction [7, 8], but they typically operate on shorter inputs and do not directly address long-horizon synthesis across many encounters. More recent LLM-based methods have improved summarization of complex clinical documents, including discharge summaries, radiology reports, and physician notes [4], though most still frame summarization as a single-pass compression problem. This limitation is important in longitudinal EHR settings, where records span many visits and time horizons, and where long-context performance can degrade in noisy clinical data. Recent work has therefore emphasized temporal reasoning in clinical language models [5]. For example, CIIN-SUMM treats notes as sequential observations to track patient state over time [6], while other approaches use timeline extraction or structured event representations, often at the cost of propagating upstream extraction errors. Outside medicine, recursive and incremental summarization have been proposed to manage long contexts by iteratively compressing text into evolving summaries [9]. However, these ideas remain underexplored for longitudinal EHRs, where factual consistency, temporal precision, and robustness to cumulative information loss are especially critical.

4 Approach

4.1 Problem Formulation

Electronic health records are long temporal documents consisting of heterogeneous clinical events such as notes, diagnoses, medications, labs, and procedures. We represent a patient EHR as a chronologically ordered sequence of text segments $X = (x_0, x_1, \dots, x_N)$, where each x_i is a chunk extracted from the raw XML record.

Our goal is to construct a sequence of progressively updated summaries $S = (s_0, s_1, \dots, s_N)$, where

$$s_0 = f_{init}(d, x_0), \quad s_i = f_{update}(s_{i-1}, x_i), \quad i \geq 1,$$

and d denotes a demographics block extracted from the XML header.

Each summary s_i should capture the clinically relevant information from chunks $x_0 \dots x_i$ while remaining concise and usable for clinical decision making. This formulation treats summarization as a **state update process**: rather than recomputing a summary from the full record, the model maintains a persistent representation of patient state that is updated as new evidence arrives. One of the main use cases for clinical summarization are tumor boards, which rely on an LLM-generated summary of the last progress note as the definition of a patient’s state during their tumor board consultation. However, this may not capture patient state due to a lack of clinical information and understanding of patient trajectories. Therefore, we hypothesize that our recursive summaries should prove as better representations of patient state at a time t .¹

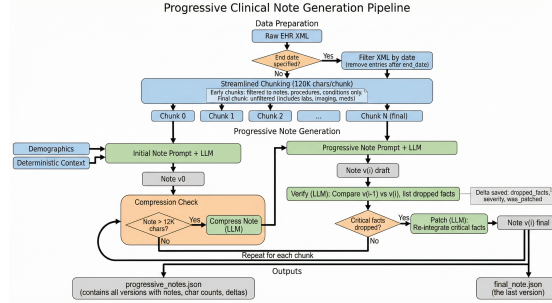


Figure 1: Flowchart showing the flow of EHR data and steps required to generate progressive notes.

4.1.1 Baseline Prompt Evaluation

We aim to evaluate multiple recursive prompt configurations, beginning with the inherited oncology-specialized prompts and introducing refined variants derived from systematic error analysis to address domain bias, temporal drift, and information erosion. Further information regarding the prompts can be found in Appendix 10.5.1 and onwards. We selected the best-performing prompt template as the instruction format for supervised fine-tuning, while retaining the existing reference summaries as gold labels.

4.2 Progressive Summarization Pipeline

4.2.1 Chunking Strategy

Each Lumia XML record is processed using a deterministic chunking pipeline. Given the full XML document D with total length $|D|$ characters, we construct chunks of at most $M = 15,000$ characters.

The pipeline consists of three steps:

1. **Final chunk reservation.** Set the final chunk boundary at position $|D| - M$. The segment $D[|D| - M :]$ is reserved as an **unfiltered chunk** containing the most recent clinical context. If $|D| \leq M$, the entire document is yielded as a single chunk and no filtering is applied.
2. **Event-type filtering.** The prefix $D[: |D| - M]$ is filtered using a regex-based procedure that matches `<event type="X">...</event>` blocks. Only events with type $\in \{\text{note, procedure, condition}\}$ are retained.
3. **Prefix sub-chunking.** The filtered prefix is split into sequential M -character segments, yielded in chronological order with an unfiltered final chunk.

This produces a sequence of chunks $(x_0, x_1, \dots, x_N)$ where x_N corresponds to the most recent unfiltered segment. At $M = 15,000$, patients produce a median of 28 chunks (range 4–253).

¹We initially aimed to use VISTA Architect for comparison, but it lacks publication and clinical validation. Since current peer-reviewed methods focus exclusively on single-pass compression, we utilize a single-summary approach as our primary baseline to ensure a clinically relevant benchmark.

4.2.2 Progressive Note Generation

At step 0 the model receives a demographics block extracted from the XML `<person>` element together with the first chunk:

`<DEMOGRAPHICS> d </DEMOGRAPHICS> || <EHR_CHUNK> x0 </EHR_CHUNK>`

and generates the initial summary s_0 using an initial-note system prompt.

For steps $i \geq 1$ the model receives the previous summary and the next chunk:

`<CURRENT_NOTE> si-1 </CURRENT_NOTE> || <NEW_SEGMENT> xi </NEW_SEGMENT>`

and produces the updated summary s_i using a progressive-update system prompt. For the final step ($i = N$), an additional instruction is appended requesting completeness verification (e.g., all treatment lines accounted for, staging reflects the most current assessment).

To control context growth, summaries exceeding a compression threshold $\tau = 12,000$ characters are compressed to approximately 0.6τ characters before the next update. The compression prompt preserves decision-relevant content (diagnoses, treatment lines, toxicities, current status) while condensing superseded imaging, routine labs, and repetitive status notes. Every intermediate summary s_i is written to disk as `note_{i}.txt`, enabling pipeline resumability after interruption—a practical necessity for patients requiring 100+ update steps.

4.2.3 Chunk Size Selection

We initially experimented with 120K-character chunks and progressive labels generated using Qwen-7B. In this configuration, later update steps required conditioning on both a long prior summary and a very large new segment. This resulted in unstable generation behavior.

Reducing chunk size to 15K characters decreased the average step input length to approximately 22K characters and increased the number of supervision points per patient (median 28 updates per patient compared with 5–6 updates under 120K chunking). However, due to API limitations, we conduct downstream predictive analyses on the 120k dataset. Consequently, our clinical quality and retention metrics reflect the optimal 15K pipeline, while the downstream prediction tasks demonstrate a conservative baseline of our method under the extreme context pressure of 120K chunks

4.3 Retention Auditing

A key challenge in progressive summarization is **information erosion**. Facts present in earlier summaries may disappear in later updates. To measure and mitigate this, we introduce a two-stage retention audit.

Verification step Given summaries s_{i-1} and s_i , an LLM verifier receives both as `<PREVIOUS_NOTE>` and `<UPDATED_NOTE>` and identifies facts present in s_{i-1} but missing or materially altered in s_i . Each dropped fact is assigned a severity label:

- **Critical:** facts that would affect the next treatment decision (diagnoses, staging, molecular profile, treatment lines, toxicities constraining future options, allergies, code status).
- **Minor:** contextual information unlikely to change the treatment plan (superseded imaging details, prior performance status, verbose tolerability descriptions).

The verifier outputs structured JSON: `{"dropped_facts": [{"fact": "...", "severity": "critical|minor", "section": "..."}]}`.

Repair step If any facts with `severity = "critical"` are detected, a repair prompt receives the current summary and the dropped facts formatted as a bulleted list with section annotations (e.g., `[Treatment Timeline] ...`). The repair integrates the facts into the relevant sections of the note rather than appending an external list.

Each verify-and-patch cycle produces a `delta_{i}.json` file recording the list of dropped facts, critical and minor counts, whether patching occurred, and timing information. These delta files constitute the structured telemetry used for all retention metrics reported in Section 5.

This mechanism acts both as a quality-control system and as a telemetry mechanism for measuring information loss over long update sequences. We note that this formulation of recursive summarization, verification, and retention auditing with unstructured EHR is one of our main novel contributions. However, the data pipeline EHR chunking was created by the VISTA team, but remains proprietary and thus code cannot be shared.

5 Experiments

5.1 Dataset

We use the MedAlign dataset, which contains de-identified longitudinal EHR records in Lumia XML format. The dataset includes 275 patients with record sizes ranging from 50KB to 3.8MB and spanning multiple clinical domains including oncology, psychiatry, pediatrics, autoimmune disease, and primary care. Using the progressive pipeline described above, we generated progressive summary labels for 241 patients using EHR chunks of 15k characters. This produces a dataset of 9,962 examples where each example corresponds to one progressive update step.

Statistic	Value
Total training examples	9,962
Progressive update examples	9,721
Median chunks per patient	28
Avg input chars (step 0)	15,450
Avg input chars (step 1+)	22,022

Table 1: Summary statistics of the progressive summarization dataset (n=241).

5.2 Training Setup

The progressive pipeline uses GPT-based models for note generation, compression, and retention auditing. Clinical summary quality is evaluated using Gemini 2.0 Flash as an LLM judge. Pipeline execution is sequential within each patient trajectory because each step depends on the previous summary. Patients are processed in parallel batches subject to rate limits, with LLM parameters outlined in Table 7.

5.3 LoRA Fine Tuning

We fine-tuned parameter-efficient LoRA adapters [10] for progressive clinical summarization, training instruction-tuned causal language models to generate step-wise summary updates from longitudinal EHR inputs. Silver-labeled JSONL data with `patient_id`, `step`, `input`, and `label` were split at the patient level to avoid leakage and reformatted into chat-style supervision with step-dependent system prompts. At `step = 0`, the model generated an initial summary from the first EHR segment; at `step > 0`, it updated the summary using a progressive prompt and new context. We trained Qwen-family models, including Qwen2.5-7B-Instruct and Qwen2.5-3B-Instruct, with LoRA applied to core projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`) using TRL SFT and PEFT. At inference, we loaded the base model plus saved adapter and generated summaries with the same step-aware prompt schema. Evaluation combined automatic overlap metrics with LLM-as-judge rubric scoring. A key implementation finding was that long, noisy XML inputs created substantial token pressure and structural degradation, making smaller chunks, context budgeting, and prompt consistency important for stable outputs.

5.4 Evaluation

We evaluate the system along three dimensions: **Clinical quality**, **information retention**, and **downstream predictive utility**.

Method	Prov.	Acc.	Thor.	Useful	Org.	Compr.	Succ.	Synth.	Overall
Prompt 1	4.750	4.783	3.517	4.117	4.583	4.933	4.633	3.903	4.403
Prompt 2	4.817	4.833	4.817	5.000	4.967	5.000	4.697	4.470	4.825
Prompt 3	4.900	4.916	4.317	4.766	4.867	4.967	4.717	4.783	4.779
120k Chunk Aggregate	4.787	4.845	3.832	4.484	4.725	4.919	4.699	4.387	4.585
B1: Raw Last Note	5.000	5.000	4.446	4.519	4.736	4.987	4.680	2.307	4.459
B2: LLM Summary of Last Note	4.661	4.717	2.309	2.848	3.526	4.843	4.422	2.745	3.759

Table 2: Custom evaluation results across progressive prompt variants (n=10), the aggregate 120k setting (n=273), and two single-note baselines. B1 is the raw last progress note(n=231); B2 is an LLM summary of the last progress note using the current single-note summarization workflow (n=231). Prov. = provenance, Acc. = accuracy, Thor. = thoroughness, Org. = organization, Compr. = comprehensibility, Succ. = succinctness, and Synth. = synthesis. Note that for B1, we expect perfect scores for Prov. and Acc. since no information was lost. Thus, this metric only serves as a reinforcement of our evaluation methodology.

Clinical quality. We use the clinically-validated PDSQI-9 rubric from EPIC, evaluating provenance, accuracy, thoroughness, usefulness, organization, comprehensibility, succinctness, and synthesis [1]. The LLM judge receives the previous summary, the raw EHR chunk, and the updated summary.

Information retention. Retention auditing logs dropped facts at each progressive step and aggregates corpus-level statistics such as critical drop rate and patch success rate.

Downstream predictive utility. To test whether summaries preserve predictive signal, we perform cancer prediction using summaries at time $t - 1$ days relative to the initial cancer diagnosis. We evaluate with two methodologies; embedding and zero-shot learning. For the embedding method, the summary is encoded with BioMed ClinicalBERT to produce an embedding, and a classifier predicts future cancer events. The zero-shot tasks uses the note as input for an LLM, which is asked to classify future cancer risk.

6 Results

6.1 Summary Quality

Table 2 reports PDSQI-9 scores for the GPT-5 progressive pipeline evaluated across multiple prompts and number of patients. The pipeline achieves strong provenance (4.84) and accuracy (4.88), indicating high traceability and low fabrication. **Thoroughness** is the weakest dimension (4.04), consistent with the information erosion inherent to long update chains. Synthesis (4.57) suggests that summaries move beyond fact listing toward clinically useful abstraction, though there remains room for improvement.

6.2 Fine-Tuned Model Results

Figure 2 shows the score distributions for Fine-tuned Model 1 and Fine-tuned Model 2 under the same evaluator setup. Both distributions are concentrated in the lower score range, indicating that weak average performance is broadly distributed rather than driven by a small number of outliers. Fine-tuned Model 2 shifts the distribution slightly relative to Model 1, consistent with incremental but limited gains. Overall, the fine-tuned models remain weak on decision-critical summary quality, suggesting that improvements in data conditioning, XML cleanup, prompt-template consistency, and decoding control are likely more important than simply training longer.

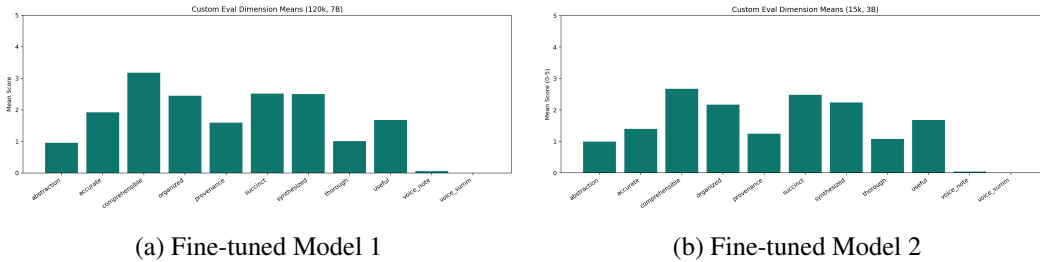


Figure 2: Distribution of evaluator scores for two fine-tuned models on the held-out set.

6.3 Retention Metrics

Table 3 reports retention audit statistics aggregated across 241 patients and 9,705 verify-and-patch steps (15K granularity). Critical fact drops occur at 8.15% of update steps. The patch mechanism triggers at 6.58% of steps and successfully repairs 80.8% of critical drops. Minor drops are pervasive (mean 3.17 per step, 127.7 per patient), reflecting the model’s tendency to condense older details during progressive updates.

Metric	Value
Verify steps	9,705
Critical facts dropped	791
Minor facts dropped	30,778
Critical drop rate (per step)	8.15%
Minor drop rate (per step)	3.17 facts
Patch trigger rate	6.58%
Patch success rate	80.8%
Per-patient minor drops (median)	81
Per-patient critical drops (median)	1

Table 3: Retention audit statistics across 241 patients (15K chunking).

6.4 Downstream Prediction

We evaluated four input formulations for a binary cancer prediction task on a matched cohort of $n = 230$ patients, including 27 patients with cancer and 203 without cancer. For patients with cancer, the prediction input was taken from the period immediately preceding diagnosis. For patients without cancer, we selected a note at the median proportional point of the longitudinal EHR timeline corresponding to the observed diagnosis point among cancer cases. This matching procedure was intended to reduce confounding from differences in EHR trajectory length.

The evaluated inputs were: **B1**, the last progress note prior to cancer diagnosis; **B2**, an LLM-generated summary of that note; and **A**, our proposed progressive summary method. We report area under the receiver operating characteristic curve (AUROC) and area under the precision-recall curve (AUPRC). Cohort imbalance ($27/230 = 11.7\%$ positive prevalence) makes AUPRC especially informative.

6.5 Matched-Cohort Results

Table 4 summarizes results for both zero-shot and embedding-based prediction settings.

Setting	Method	AUROC	AUPRC
Zero-shot	B1	0.7153	0.2965
	B2	0.6558	0.2539
	A	0.8054	0.3422
Embedding	B1	0.6121	0.2289
	B2	0.6355	0.1614
	A	0.6566	0.2152

Table 4: Matched-cohort cancer prediction performance. B1 is the last progress note prior to diagnosis, B2 is an LLM summary of that note, and A is our proposed method. (n=241)

7 Analysis

Progressive summarization reveals a tradeoff between conciseness and retention. Minor fact drops average 3.17 per step, or about 85 for a median-length patient trajectory, and likely contribute to thoroughness being the weakest PDSQI-9 dimension. Information erosion is most pronounced in complex cases: patients with more than 100 chunks have the highest drop rates, and one patient accumulated 756 minor drops. Compression is the dominant source of information loss, with critical drops clustering around compression events and clinically dense transitions, suggesting that compression-aware auditing could improve patch coverage. Chunk size also affects stability. Larger chunks produce sharper context shifts between updates, whereas the 15K scheme yields smaller, more stable transitions. At this granularity, each update incorporates roughly 4K tokens of new information,

while the combined input length stays within a more stable range for both proprietary and open-weight models. Fine-tuning provided modest gains in fluency and conciseness but did not reliably preserve oncology decision-critical content under long, noisy inputs. The primary failure mode was **context pressure**: even with long-context base models, usable context remained effectively bounded, leading to truncation of noisy XML inputs and weaker coverage of clinically important details. A second issue was **input noise**, as repeated boilerplate and low-signal XML content encouraged extractive and repetitive outputs.

Zero-shot vs. fine-tuned comparison While our primary retention pipeline relied on proprietary models, we also investigated whether open-weights could replicate this structure. We compared base-model zero-shot outputs against LoRA fine-tuned outputs on four held-out examples. Across these samples, both zero-shot and fine-tuned generations remained largely extractive and often reflected raw XML/EHR phrasing rather than the intended structured summary format. The reference labels consistently followed the four-section template (Patient & Diagnosis, Treatment Timeline, Current Disease Status, Decision-Relevant Factors), whereas model outputs did not reliably reproduce this structure. Fine-tuned outputs were generally longer, but the added length did not translate into better structural adherence or clearer clinical summaries in this small qualitative check.

Downstream Prediction Analysis. Method A consistently achieved the highest AUROC in both zero-shot (0.8054) and embedding (0.6566) settings, outperforming B1 and B2 and demonstrating strong preservation of discriminative signals. However, AUPRC results were more nuanced: while A led the zero-shot setting (0.3422), the raw last-note baseline (B1) achieved the highest AUPRC in the embedding setting (0.2289). Consistently, **B2 did not provide a reliable advantage**, generally underperforming B1 across most metrics and settings. Overall, these results indicate that summarization alone is not uniformly beneficial; effective longitudinal methods must carefully preserve fine-grained predictive evidence alongside global discrimination.

We note two key limitations of our results: our limited dataset size and unbalanced task. Due to data access constraints, we were unable to access a larger, more oncologically-focused dataset. This would have improved the reliability of our results and enabled a more diverse set of tests, such as treatment and outcome prediction. Despite these limitations, this framework establishes a clear baseline for the utility of recursive summarization. Future work will expand this approach to larger, oncology-specific datasets to evaluate a broader range of clinical outcomes.

8 Conclusion

We present a progressive summarization framework for longitudinal EHR data that incrementally updates a running clinical summary as new segments are processed. The system combines sequential summarization with retention auditing so that information loss becomes measurable and partly correctable. On the GPT-5 pipeline, the method achieves provenance 4.84/5 and accuracy 4.88/5 on PDSQI-9, while the retention audit detects critical fact drops at 8.15% of update steps and repairs 80.8% of them. In the downstream cancer prediction task, progressive summaries achieve the highest AUROC (0.8054 in the zero-shot setting) among the evaluated input formulations.

The resulting dataset of 9,962 progressive examples across 241 patients provides a foundational resource for future work in training open-weight models to navigate long clinical histories and preserve decision-relevant information. More broadly, our findings suggest that longitudinal clinical summarization should be treated as a state-tracking problem, with explicit mechanisms for retention and repair rather than simple compression alone.

9 Team contributions

All three team members contributed across prompt design, experiments, analysis, and writing. Eashan designed and implemented the progressive note generation pipeline and led the downstream prediction experiments. Asmita prepared the chunked and labeled dataset for LoRA fine-tuning, worked on the LLM-as-a-judge evaluation pipeline, and co-wrote the report. Arlina conducted the Qwen fine-tuning experiments, analyzed the fine-tuned model outputs, and co-wrote the report. All three team members also jointly developed the prompts for progressive summarization and contributed to overall project direction and experimental decisions.

References

- [1] Emma Croxford, Yanjun Gao, Nicholas Pellegrino, Karen K. Wong, Graham Wills, Elliot First, Miranda Schnier, Kyle Burton, Cris G. Ebby, Jillian Gorskic, et al. Development and validation of the provider documentation summarization quality instrument for large language models. *arXiv preprint arXiv:2501.08977*, 2025.
- [2] George Hripcsak and David J Albers. Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association*, 2013.
- [3] Benjamin Shickel et al. Deep ehr: A survey of recent advances in deep learning techniques for electronic health record analysis. *IEEE Journal of Biomedical and Health Informatics*, 2017.
- [4] Dave Van Veen, Charlotte Van Uden, Louisa Blankemeier, et al. Adapted large language models can outperform medical experts in clinical text summarization. *Nature Medicine*, 30:1134–1142, 2024.
- [5] Maya Kruse, Shiyue Hu, Nicholas Derby, Yifu Wu, Samantha Stonbraker, Bingsheng Yao, Dakuo Wang, Elizabeth Goldberg, and Yanjun Gao. Large language models with temporal reasoning for longitudinal clinical summarization and prediction. 2025.
- [6] V D’Souza et al. Clin-summ: Temporal summarization of longitudinal clinical notes. *medRxiv*, 2025.
- [7] Emily Alsentzer, John Murphy, et al. Publicly available clinical bert embeddings. *Proceedings of Clinical NLP Workshop*, 2019.
- [8] Jinhyuk Lee et al. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 2020.
- [9] Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models. *arXiv preprint arXiv:2308.15022*, 2023.
- [10] 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. *arXiv preprint arXiv:2106.09685*, 2021.

10 Appendix

10.1 Dataset Schema

Each training example in `training_dataset_15k.jsonl` contains:

```
{
  "patient_id": "124930886",
  "step": 0,
  "total_steps": 7,
  "input": "<DEMOGRAPHICS>\n...\n</DEMOGRAPHICS>\n\n<EHR_CHUNK>\n...\n</EHR_CHUNK>",
  "label": "## Patient & Diagnosis\n...",
  "input_chars": 15485,
  "label_chars": 598
}
```

The input field is constructed differently for initialization and update steps. At step 0, demographics are extracted from the XML `<person>` element and provided as context:

```
if step == 0:
    parts = []
    if demographics:
        parts.append("<DEMOGRAPHICS>\n"
                    + demographics
                    + "\n</DEMOGRAPHICS>\n")
    parts.append("<EHR_CHUNK>\n"
                + chunks[step]
                + "\n</EHR_CHUNK>")
    input_text = "\n".join(parts)
```

For steps ≥ 1 , the input concatenates the previous step's label (the running summary) with the next chunk:

```
else:
    prev_note = notes[step - 1]
    input_text = (
        "<CURRENT_NOTE>\n" + prev_note
        + "\n</CURRENT_NOTE>\n\n"
        + "<NEW_SEGMENT>\n" + chunks[step]
        + "\n</NEW_SEGMENT>"
    )
```

For labels, each entry in the `notes` array stores the summary produced after processing the corresponding chunk. The dataset builder re-chunks the source XML with the same deterministic `iter_chunks()` function used during label generation; chunk counts and byte-level sizes were verified to match exactly for all 241 patients.

10.2 Chunking Algorithm

The `iter_chunks` function implements the streamlined chunking strategy described in Section 3. The core logic is:

```
def iter_chunks(xml_text, max_chars=15000,
                min_final_chunk_ratio=0.8):
    if len(xml_text) <= max_chars:
        yield xml_text
        return

    final_start = len(xml_text) - max_chars
```

```

beginning = xml_text[:final_start]
final_chunk = xml_text[final_start:]

filtered = filter_xml_events(
    beginning,
    {"note", "procedure", "condition"}
)

for i in range(0, len(filtered), max_chars):
    chunk = filtered[i : i + max_chars]
    if chunk.strip():
        yield chunk

yield final_chunk

```

The event filter operates via regex over XML event blocks, retaining events whose type attribute is in the keep set:

```

pattern = r'(<event\s+[^\>]*type\s*=\s*'
        r'["\'](\w+)["\']\s*>)(.*?)(</event>)'
filtered = re.sub(pattern, keep_fn, xml_text,
                 flags=re.DOTALL)
filtered = re.sub(r'<entry[^\>]*>\s*</entry>', '',
                 filtered, flags=re.DOTALL)

```

This ensures that older portions of the record are reduced to high-signal events while the most recent portion is preserved in full.

10.3 Retention Audit Internals

The verify-and-patch mechanism runs after every update step ($i > 0$). The verifier LLM call receives `<PREVIOUS_NOTE>` and `<UPDATED_NOTE>`. Its JSON response is parsed with a regex extractor that handles `</think>` tags (for reasoning models) and markdown code fences:

```

text = re.sub(r"````(?:json)?\s*", "", text)
m = re.search(r"\{[\s\S]*\}", text)
parsed = json.loads(m.group(0))
facts = parsed["dropped_facts"]

```

If any facts have `severity == "critical"`, the patch prompt receives the dropped facts formatted with section annotations:

```

facts_text = "\n".join(
    f"- [{f['section']}] {f['fact']}"
    for f in critical
)

```

Each step produces a `delta_{i}.json` file recording:

```

{
  "dropped_facts": [...],
  "critical_dropped": 0,
  "minor_dropped": 3,
  "was_patched": false,
  "verify_time": 14.9
}

```

When patching occurs, additional fields record `pre_patch_chars`, `post_patch_chars`, and `patch_time`. These delta files are aggregated to produce all retention statistics reported in Section 5.

10.4 Compression Mechanism

When a summary exceeds the threshold $\tau = 12,000$ characters, the compression step targets $0.6\tau \approx 7,200$ characters. The compression prompt specifies a hierarchy of operations:

- **Preserve** (never remove): diagnosis, histology, molecular profile, staging, each treatment line with regimen/dates/response, toxicities constraining future options, current disease status, allergies, contraindications.
- **Condense** (reduce detail but keep the fact): older imaging collapsed to “stable disease [date range]”, treatment tolerability summarized as “tolerated well” or “dose-reduced for [toxicity]”, routine labs omitted unless documenting organ dysfunction.

Section structure is maintained through compression.

10.5 Prompt Structure

The system uses four prompt templates loaded from text files at runtime:

- **Initial note** (`initial_note.txt`): System prompt instructing patient-context detection (oncology / general clinical / pediatric) and format selection. Specifies decision-relevant information to extract: diagnoses, medications with doses and dates, treatment histories as discrete entries, standardized symptom scale trajectories, psychiatric safety events, cardiovascular baselines, immunosuppression timelines, infectious disease screening, growth data (pediatric), vaccination histories, and pending actions.
- **Progressive update** (`progressive_note.txt`): Defines four action categories: INCORPORATE (new findings), UPDATE IN PLACE (superseded status fields), PRESERVE (active diagnoses, treatment trials, scale trajectories, baselines, screening dates, pending actions), and CONDENSE (superseded imaging, routine stable labs). Includes a **cumulative-drop guardrail**: before omitting any fact, the model checks whether the note has already lost detail in that area across prior versions. Enforces date accuracy: documented dates must not be altered during condensation.
- **Retention audit** (`note_verify.txt`): Defines critical vs. minor severity criteria. Instructs the verifier to only flag facts from the previous note that are missing in the updated note, and not to flag correct updates or stylistic changes.
- **Patch** (`note_patch.txt`): Instructs natural re-integration of dropped critical facts into the appropriate note sections. Prohibits appending a separate “restored facts” block.

10.5.1 Prompt Templates

Baseline prompts (Sections 10.6–10.9) were inspired from current tumor board prompts for summarization and illustrate the oncology-specialized framing we generalize beyond. Unified improved prompts (Sections 10.10–10.11) incorporate lessons derived from analysis of the full 11-patient pilot cohort.

10.6 Baseline Initial Note Prompt

You are a senior oncology fellow creating a clinical summary from electronic health records. Your goal is to capture information that would inform the next treatment decision for this patient’s malignancy.

This is EHR SEGMENT 1 of a multi-part record. You are building a progressive summary that will be updated as more segments arrive.

DECISION-RELEVANT INFORMATION - include if present:

- Cancer type, histology, grade, pathology details
- Staging (clinical and pathological TNM, overall stage)
- Tumor-specific molecular profile

- Each treatment line: drugs/regimen, start/stop dates, intent, best response, reason for discontinuation
- Toxicities that led to dose modification, delay, or change
- Most recent imaging: modality, date, key findings
- Performance status (ECOG)
- Contraindications: allergies, code status, bleeding risk

EXCLUDE:

- Conditions unrelated to oncological care
- Routine non-oncological lab values
- Non-oncological medications

FORMAT:

```
## Patient & Diagnosis
## Treatment Timeline
## Current Disease Status
## Decision-Relevant Factors
```

10.7 Baseline Progressive Note Prompt

You are maintaining a clinical summary focused on oncological decision-making. You will receive:

1. CURRENT_NOTE
2. NEW_SEGMENT

Task: produce an UPDATED summary incorporating decision-relevant information from the new segment.

Maintain structure. Preserve diagnosis and treatment lines. Update staging and imaging in place. Omit superseded data. No commentary - output only the updated note.

10.8 Verify Prompt (Fact-Drop Auditor)

You are a clinical quality auditor checking whether a progressive summary update dropped decision-relevant information.

Input:

1. PREVIOUS_NOTE
2. UPDATED_NOTE

Identify facts missing or materially altered.

CRITICAL:

- Diagnosis, staging
- Treatment lines and toxicities
- Allergies, contraindications

MINOR:

- Older imaging
- Routine labs
- Social history

Return JSON only:

```
{"dropped_facts":[...]}
```

10.9 Patch Prompt (Fact Restoration)

Re-integrate critical clinical facts that were dropped.

1. Place facts in correct section.
2. Integrate naturally.
3. Keep concise.
4. No commentary.

10.10 Unified Improved Initial Prompt

You are a senior clinical fellow creating a summary from EHR data. Your goal is to capture the minimal sufficient state needed for the next clinical decision.

STEP 1: Determine context:

- Oncology
- Pediatric
- General Clinical

Capture coded diagnoses, medications, lab trajectories, symptom scales, immunosuppression history, thrombotic risk factors, screening status, pregnancy status, allergies, code status.

Preserve exact documented dates. Do not shift dates.
If unknown, mark explicitly.

End with CLINICAL SYNTHESIS:

1-2 sentences describing condition interactions and key risks.

10.11 Unified Improved Progressive Prompt

Maintain decision-relevant clinical state across updates.

Incorporate:

- New diagnoses or treatment changes
- New labs, imaging, trajectories
- Symptom scale updates
- Pregnancy or pediatric growth updates
- Screening completion or new gaps

Never drop:

- Active diagnoses
- Failed treatment history
- Allergies, code status
- Risk factors (bleeding, thrombotic, autoimmune)

Preserve original dates.

No future leakage.

Maintain structure.

End with CLINICAL SYNTHESIS.

11 Evaluation Rubric

11.1 Fact-Drop Distribution

Table 5 shows the per-patient distribution of fact drops across the 241-patient cohort.

Percentile	Minor drops	Critical drops
10th	16	0
25th	40	0
50th (median)	81	1
75th	169	3
90th	282	10
Max	756	73

Table 5: Per-patient fact-drop distribution (241 patients, 9,705 verify steps). Minor drops occur universally; critical drops are concentrated in complex patients with long records.

Parameter	Value
Note generation model	GPT-5 (200K context)
Max output tokens	32,768
Retention audit model	GPT-5
Evaluation judge model	Gemini 2.0 Flash
Chunk size (M)	15,000 characters
Compression threshold (τ)	12,000 characters
Compression target	$0.6\tau = 7,200$ characters
Event types retained (prefix)	note, procedure, condition
Min final chunk ratio	0.8
Batch parallelism	Up to 3 concurrent patients
Rate-limit retries	15 (429 errors)
Timeout retries	3

Table 6: Pipeline configuration parameters.

11.2 Pipeline Configuration

12 Pipeline Configuration

Stage	Model	Temp.	Top- p	Max output tokens	Input \$ / 1M	Output \$ / 1M	Calls / pt (6 × 120k)	Calls / pt (48 × 15k)	Cost / pt (6 × 120k / 48 × 15k)
Generation	gpt-5	N/A	N/A	32,768	1.25	10.00	11	95	0.99 / 8.55
Evaluation	gemini-2.0-flash	0.1	N/A	8,192	0.10	0.40	6	48	0.01 / 0.04

Table 7: Default code settings for `prepare_progressive` (generation) and `evaluate_progressive` (evaluation). For generation, `prepare_progressive.py` uses `gpt-5`; temperature cannot be changed because the API calls are routed through Secure-LLM, a proprietary method to use a PHI-safe API key. For evaluation, `evaluate_progressive.py` uses `temperature=0.1`; with `gemini-2.0-flash`, this is applied. Top- p cannot be changed due to the limitations of the PHI-safe API key. Note that generation uses $2C - 1$ calls for C chunks, with a variable number of calls if a critical error is detected, and evaluation uses C calls. Cost per patient is shown for the $6 \times 120k$ and $48 \times 15k$ chunk scenarios. Note that the input tokens for each model is 200k and 1M tokens for `gpt-5` and `gemini-2.0-flash` respectively.

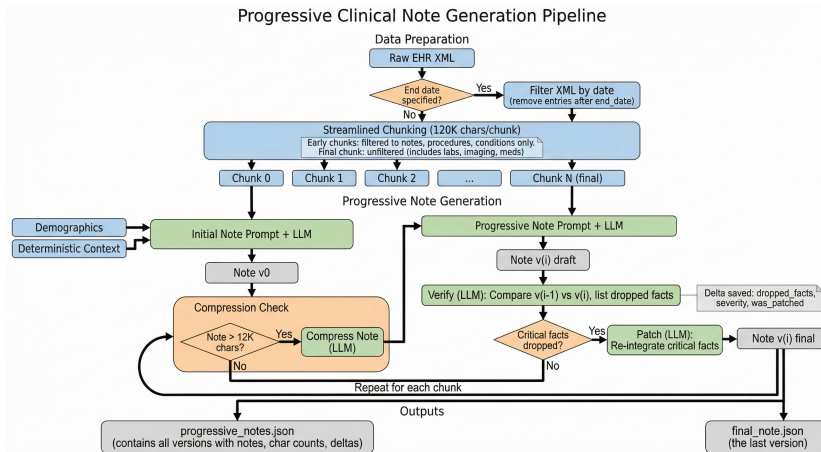


Figure 3: Flowchart showing the flow of EHR data and steps required to generate progressive notes.

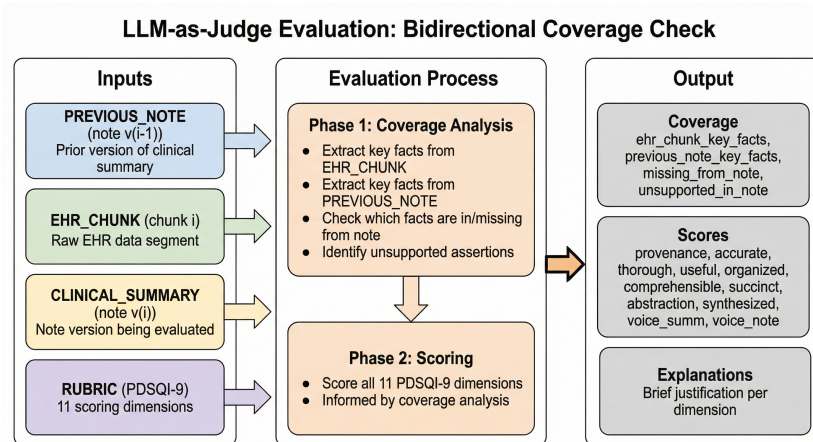


Figure 4: Flowchart showing the inputs for the LLM-as-a-judge and the evaluation scheme for each progressive note.

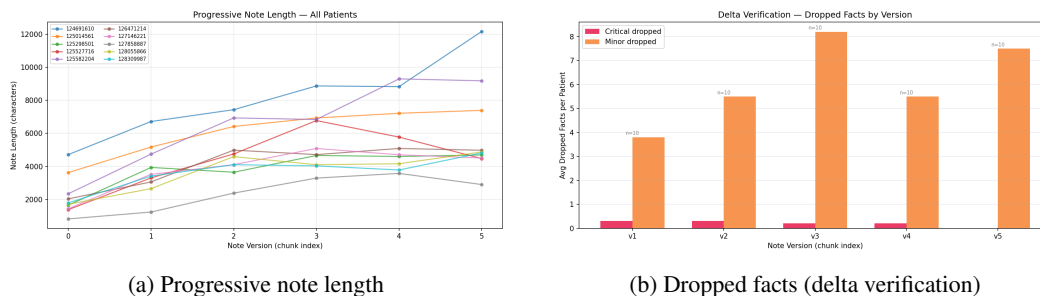


Figure 5: Longitudinal stability under the base prompt.

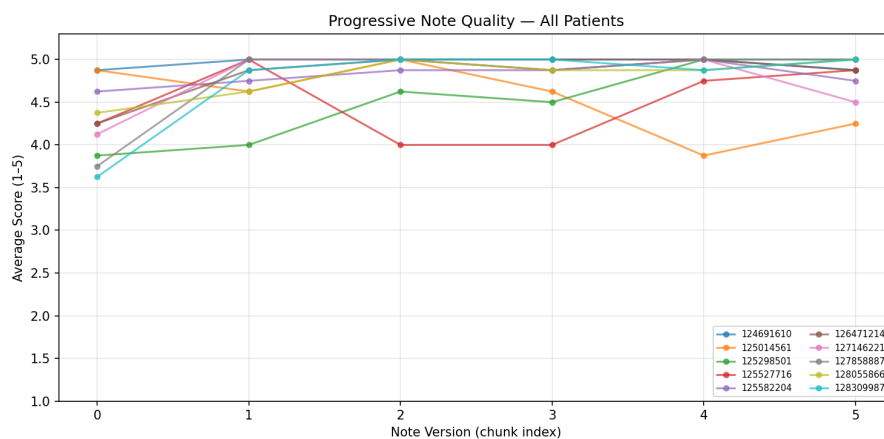


Figure 6: Progressive documentation quality across versions under base prompt.

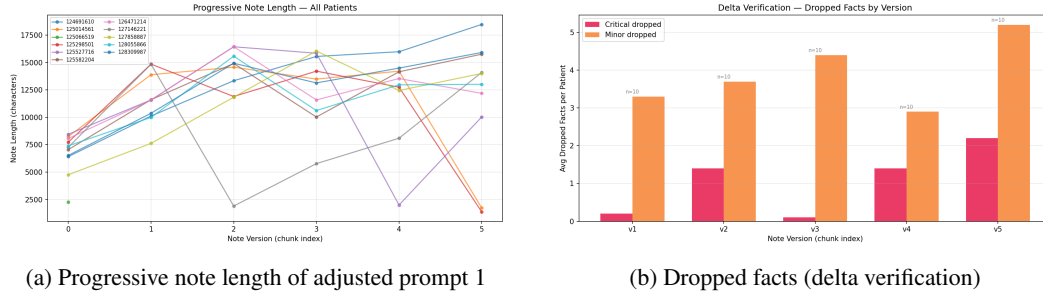


Figure 7: Longitudinal stability under the adjusted prompt 1.

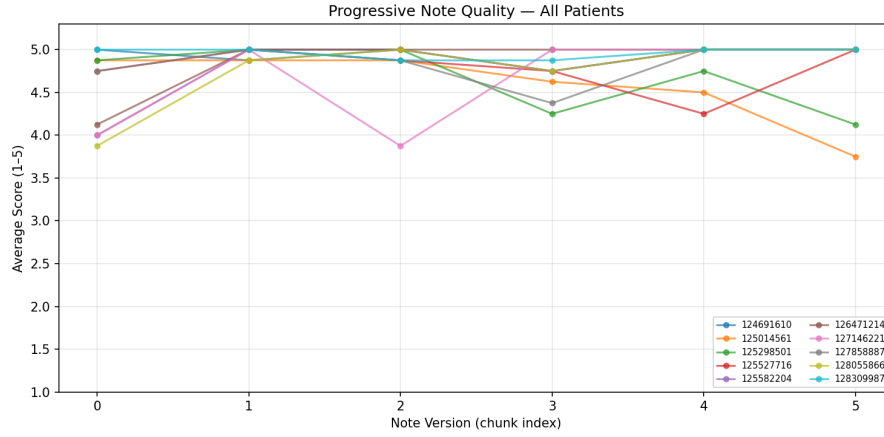


Figure 8: Progressive documentation quality across versions under the adjusted prompt 1.

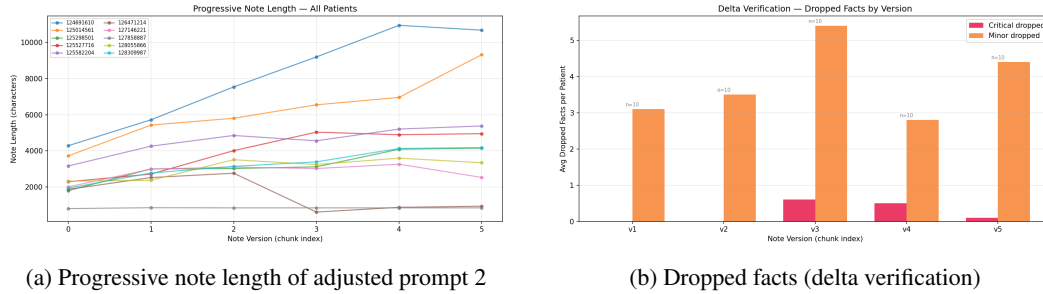


Figure 9: Longitudinal stability under the adjusted prompt 2.

Method	Prov.	Acc.	Thor.	Useful	Org.	Compr.	Succ.	Synth.	Overall
Prompt 1	4.750	4.783	3.517	4.117	4.583	4.933	4.633	3.903	4.403
Prompt 2	4.817	4.833	4.817	5.000	4.967	5.000	4.697	4.470	4.825
Prompt 3	4.900	4.916	4.317	4.766	4.867	4.967	4.717	4.783	4.779
120k Aggregate	4.787	4.845	3.832	4.484	4.725	4.919	4.699	4.387	4.585
B1: Raw Last Note	5.000	5.000	4.446	4.519	4.736	4.987	4.680	2.307	4.459
B2: LLM Summary of Last Note	4.661	4.717	2.309	2.848	3.526	4.843	4.422	2.745	3.759

Table 8: Custom evaluation results across progressive prompt variants, the aggregate 120k setting, and two single-note baselines. B1 is the raw last progress note; B2 is an LLM summary of the last progress note using the current single-note summarization workflow. Prov. = provenance, Acc. = accuracy, Thor. = thoroughness, Org. = organization, Compr. = comprehensibility, Succ. = succinctness, and Synth. = synthesis.

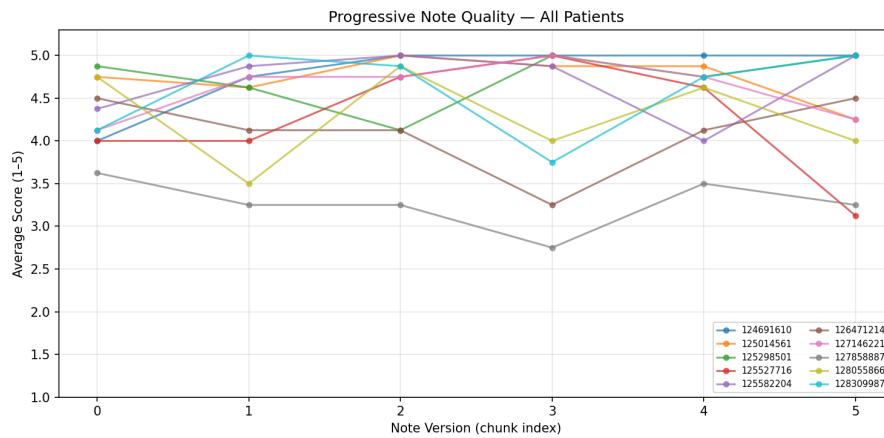


Figure 10: **Progressive documentation quality across versions under the adjusted prompt 2.**

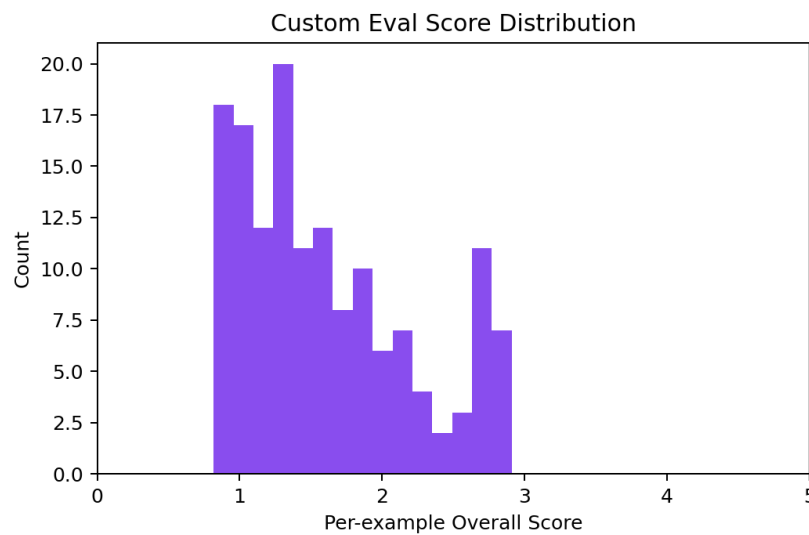


Figure 11: LLM as a Judge Scores for Finetuned Model 1 (120k characters per chunk and Qwen 7B model).

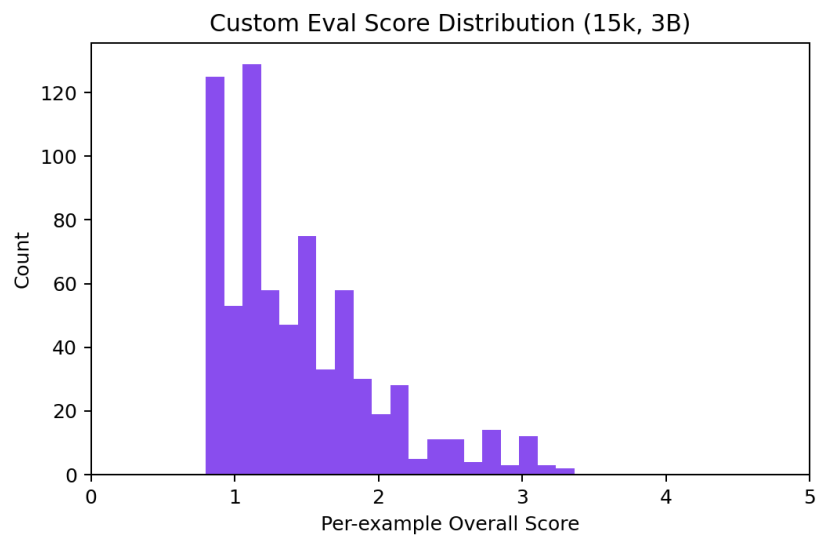


Figure 12: LLM as a Judge Scores for Finetuned Model 2 (15k characters per chunk and Qwen 3B model).