Towards Practical Application of Local LLMs in Psychiatry

Noe Javet

June 6, 2025

"The more I think about language, the more it amazes me that we understand each other at all."

— Kurt Gödel

Contents

Abstract

Large language models (LLMs) have shown strong performance in clinical text analysis but often rely on proprietary infrastructure with high resource demands. This thesis explores whether small, open-source LLMs (7–8 billion parameters) can be optimized to approximate the analytical capabilities of models like GPT-4, specifically in evaluating medical discharge summaries. A task-specific framework is developed to measure completeness, clinical relevance, and reasoning quality. We apply prompt engineering, system prompt refinement, and tool-augmented workflows to improve performance. Results show that, while smaller models have limitations in generalization and reasoning, targeted interventions significantly improve their utility, supporting the viability of resource-efficient deployment in clinical contexts.

1 Introduction

Large language models (LLMs) have rapidly advanced natural language understanding across domains such as education, law, and medicine.[1] In clinical settings, they offer new opportunities for automating tasks like summarization and document quality control. Medical discharge reports, in particular, require careful verification of structure and content completeness—tasks that demand not only linguistic fluency but also medical reasoning.

State-of-the-art proprietary systems such as ChatGPT demonstrate impressive performance in these tasks but come with significant drawbacks: high computational cost, opaque training processes, and regulatory concerns regarding data privacy. These limitations hinder their adoption in sensitive environments like healthcare. Moreover it is important to distinguish that ChatGPT is not merely a language model but a full-stack application that integrates one or more LLMs (e.g. GPT-4) with additional infrastructure, memory components, and reinforcement-based interaction layers.

Open-source LLMs in the 7–8 billion parameter range provide an attractive alternative due to their local deployability and transparency. However, their capacity to handle complex medical evaluation tasks remains insufficiently validated. This thesis investigates whether such models, when carefully adapted through prompt engineering and agentic workflows, can approach the performance of their proprietary counterparts in the domain of clinical discharge report analysis.

This thesis addresses the following research question:

> Can small, open-source language models (8B parameters) be optimized to evaluate the completeness of medical discharge reports at a level comparable to that of proprietary models such as ChatGPT?

To answer this question we focus on the task of section-level completeness analysis: identifying whether critical information (e.g., diagnoses, medication, follow-up plans) is present and appropriately described. To this end, we develop an evaluation framework that incorporates reference annotations, model-guided assessments, and performance metrics such as precision, recall, and relevance. We further explore prompt engineering techniques, instruction design, and lightweight agentic workflows to enhance model behavior without fine-tuning.

2 Methods

To evaluate the capabilities of various language models (LLMs) in processing medical discharge reports, a multi-stage evaluation pipeline was implemented. The process consisted of document preprocessing, prompt engineering, model inference, and result evaluation using defined metrics.

2.1 Dataset

The dataset used for evaluation consists of seven medical discharge reports. Among them, two reports are considered complete: one includes section-level annotations and one does not.

Report	**Missing Sections**
Report 1	Medical History
Report 2	Substance use history
Report 3	Psychopharmacology
Report 4	Clinical course
Report 5	Diagnoses

Table 1: Overview of missing sections in the five incomplete discharge reports.

Report 1 and 2 contain section headers without meaningful content, while Report 3 and 4 do have valid german statements, but without any information. Report 5 does have a valid diagnosis, but lacks the structured format or specificity typically required for clinical documentation.

2.2 Evaluation Metric

To assess the performance of the language models in identifying incomplete sections within medical discharge reports, we define the following evaluation categories:

- **True Positive (TP)**: The model correctly identifies a section as incomplete when it is indeed missing or insufficient according to the reference annotation.
- **False Negative (FN)**: The model incorrectly classifies an incomplete section as complete, thereby failing to detect the omission.
- **Irrelevant (IR)**: The model output diverges from the prompt instructions, such as by hallucinating content, introducing unrelated information, or failing to perform the evaluation task.

2.3 Language Model Configuration

For all model evaluations, a deterministic generation configuration was used to promote consistency and minimize variance across runs. Specifically, we selected a low temperature along with constrained sampling parameters.

Context Window (CTX)	Temperature	top_k	top_p
32768	0.0	8	0.4

Table 2: LLM Settings

These settings reduce randomness in token selection and encourage the model to produce concise, focused outputs aligned with the most probable completions. The rationale behind this choice is twofold. First, clinical report evaluation is a task that prioritizes factual precision and structural adherence over creativity or linguistic diversity. Second, high-temperature outputs tend to increase hallucination rates and irrelevance, which directly impacts the reliability of model judgments in a safety-critical domain such as medicine. By enforcing a narrower decoding distribution, we ensure that the model's responses remain stable and reproducible, which is essential for systematic error analysis and metric comparison across different prompt configurations. Each model was invoked with a system message tailored to the specific evaluation task. For instance, when assessing section completeness, the system message defined what constitutes a complete diagnosis section, how to handle missing or ambiguous information, and what format the response should follow (e.g., JSON structure indicating completeness).

2.4 Context Length

The context length (`num_ctx`) of a language model defines how many tokens it can process in a single prompt. This parameter is critical for tasks involving long documents, such as medical discharge reports. Models like LLaMA 3.1 and Mistral are typically pretrained with maximum context lengths of 8192 or 32768 tokens. However, many runtime environments (e.g., `Ollama`) set lower default values for performance and stability reasons.

In this study, initial evaluations were conducted with the default setting of `num_ctx` = 2024, as configured by `Ollama`'s default Modelfile. Since this was insufficient to fully encode several documents, a custom `Modelfile` was created to raise the limit to 32768 tokens. The effective context length depends not only on the model architecture but also on implementation and available GPU memory. All models except `llama3.1:8b-instruct-q8_0` were executed using 4-bit quantization on an RTX A2000 GPU (8GB VRAM).

2.5 Software Architecture

To support flexible experimentation with different prompting configurations, model parameters, and input formats, a minimal web-based application was developed. The system consists of a Python backend built with `FastAPI` and a lightweight frontend implemented using `Vue.js`. Communication between the frontend and backend is handled via asynchronous HTTP requests, enabling low-latency interactions with local language models served through the Ollama interface.

The backend is responsible for handling document uploads, formatting inputs (e.g., converting DOCX to plain text or markdown), selecting models, applying system prompts, invoking the language model, and collecting outputs. The frontend provides a user-friendly interface that allows users to select evaluation modes, inspect model responses, and track performance metrics in real time.

This architecture facilitates experimentation not only for technical users but also for non-technical stakeholders, such as clinicians or researchers, who may want to test how different models or prompts behave on actual medical documents. By abstracting away the complexity of model invocation and formatting logic, the system enables interactive exploration of language model behavior in a controlled and reproducible environment.

2.6 Evaluation

The following language models were evaluated in this study with observed VRAM and RAM usage based on `nvidia-smi` and Linux `htop`

Model	**Parameter Size**	**VRAM (GB)**	**RAM (GB)**
Command-r7b	7B	6.5-7.5	4
Hermes 3 (LLaMA 3.1 base)	8B	6.5-7.5	4
LLaMA 3.1	8B	6.5-7.5	4
LLaMA 3.1 Instruct (Quantized, q8_0)	8B	6.5-7.5	4
Mistral Instruct	7B	6.5-7.5	4

Table 3: Evaluated local language models and their parameter sizes.

Initially, raw discharge summaries in DOCX format were parsed using the `python-docx` library. This stage focused on extracting plain text content from the original documents while preserving section headers and structural elements. The extracted text was then sent to the LLMs using the following system prompt:

```
Du bist ein Evaluator für medizinische Austrittsberichte einer
Psychiatrie. Prüfe den Austrittsbericht auf fehlende medizinische
Informationen und gib **ein reines JSON-Objekt** nach
folgendem Schema aus:

* ‘"Abschnittsname":
0‘ wenn es Evidenz gibt, dass der Abschnitt medizinisch vollständig ist
* ‘"Abschnittsname":
1‘ wenn nicht

Gib das JSON-Objekt OHNE Kommentare, Fließtext oder Einleitung zurück.
```

Subsequently, to reduce variability introduced by inconsistent formatting, the documents were manually reformatted to adhere to a markdown-like structure. These manually structured texts were then re-evaluated using the same baseline system prompt to measure improvements attributable solely to input clarity.

In the final stage, the same markdown-formatted texts were reprocessed using an agentic workflow with access to a tool. The system prompt for the agentic workflow was modified to include the tool use:

```
Du bist ein Evaluator für medizinische Austrittsberichte einer
Psychiatrie. Prüfe den Austrittsbericht auf fehlende medizinische
Informationen und gib **ein reines JSON-Objekt** nach
folgendem Schema aus:

* ‘"Abschnittsname":
0‘ wenn es Evidenz gibt, dass der Abschnitt medizinisch vollständig ist
* ‘"Abschnittsname":
1‘ wenn nicht

Beim Abschnitt ’Diagnosen’ kannst du das Tool
‘analyze_diagnosis(text: str)‘ verwenden.
Nutze es nur dann, wenn du dir unsicher bist, ob der
Abschnitt vollständig ist.

Gib das JSON-Objekt OHNE Kommentare, Fließtext oder Einleitung zurück.
```

If the model decides to use the tool, the Agent then calls another LLM with the following system prompt:

```
Du bist ein Spezialist für psychiatrische Diagnosen.

Beurteile, ob ein Diagnosen-Abschnitt vollständig ist.

Ein Abschnitt ist **vollständig**, wenn:
* Mindestens ein ICD-10-Code wie "F01" oder "F10.1" enthalten ist.
* Die Diagnosebeschreibung ist medizinisch konsistent mit dem ICD-Code.

**Diagnosebezeichnungen ohne ICD-10-Codes gelten als unvollständig.**

Antwortformat:

Wenn vollständig:
{"Diagnosen": 0}

Wenn unvollständig:
{"Diagnosen": 1}

Antwort ausschließlich im JSON-Format. Keine Kommentare. Kein Fließtext.
```

3 Experimental Setup

All experiments were conducted locally on a Lenovo ThinkPad P1 Gen 7 laptop. The system is equipped with an Intel Core i7-12800H CPU (14 cores, 20 threads), 64 GB of DDR5 RAM, and an NVIDIA RTX A2000 Laptop GPU with 8 GB of VRAM. The machine runs Arch Linux with a minimal configuration optimized for reproducibility and resource monitoring.

Local language model inference was managed using the Ollama framework, which allows for efficient execution of quantized and full-precision models on consumer-grade hardware. Model-specific context limits and system prompts were configured via custom `Modelfile` definitions. Execution times were measured using Python's `time` module during end-to-end processing of each document, including prompt formatting and response parsing.

This setup enables the deployment and evaluation of 7–8B parameter models under real-world constraints, simulating a practical local inference scenario without access to high-end server hardware or cloud acceleration.

4 Results

4.1 Evaluation Setups

Setup A: Baseline prompt applied to DOCX reports converted to plain text using `python-docx`. **Setup B**: Baseline prompt applied to manually formatted markdown versions of the same reports. **Setup C**: Agentic prompting framework that invokes tool-calling for diagnosis section analysis.

4.2 Quantitative Metrics

The best possible outcome is 5 points for true positives, 0 points for false negatives and 0 points for irrelevant.

Figure 1 shows the distribution of evaluation outcomes (TP, FN, IR) for setup A

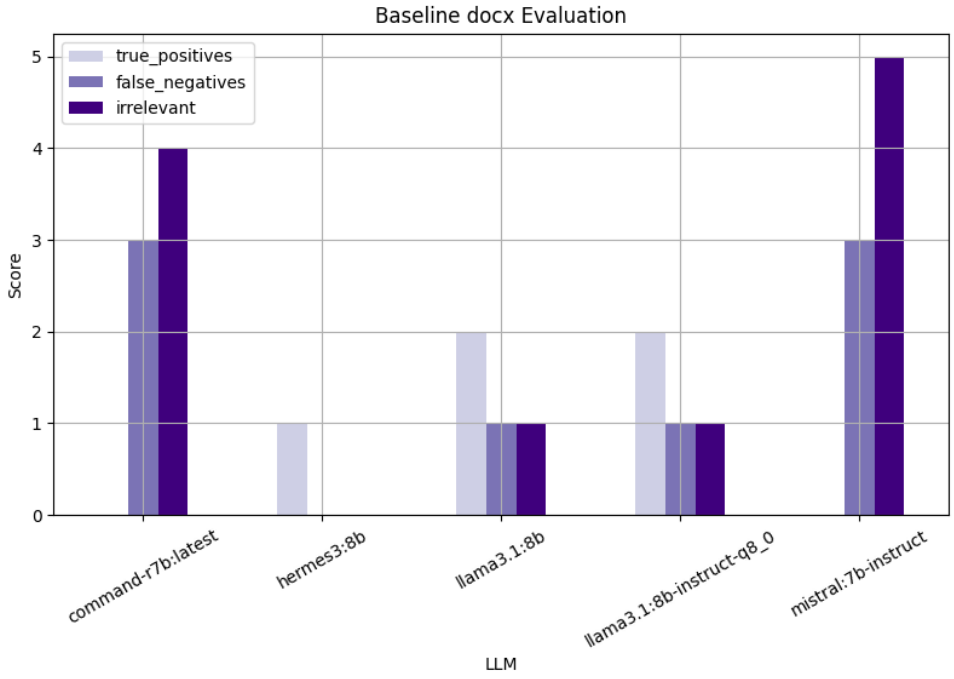

Figure 1: Evaluation outcome counts for setup A.

Figure 2 shows the distribution of evaluation outcomes (TP, FN, IR) for setup B

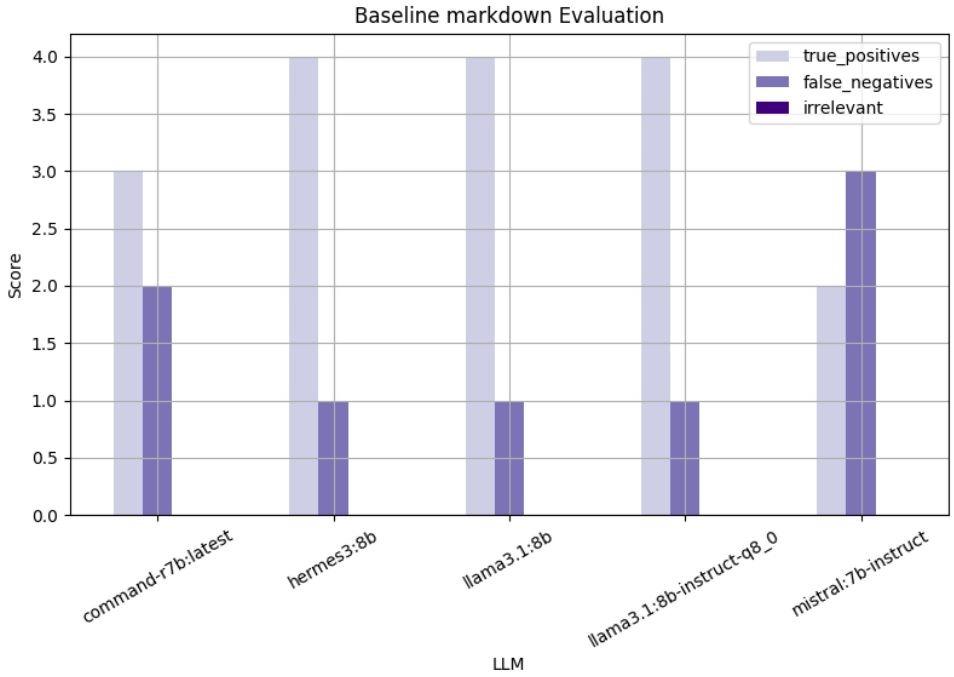

Figure 2: Evaluation outcome counts for setup B.

Figure 3 shows the distribution of evaluation outcomes (TP, FN, IR) for setup C

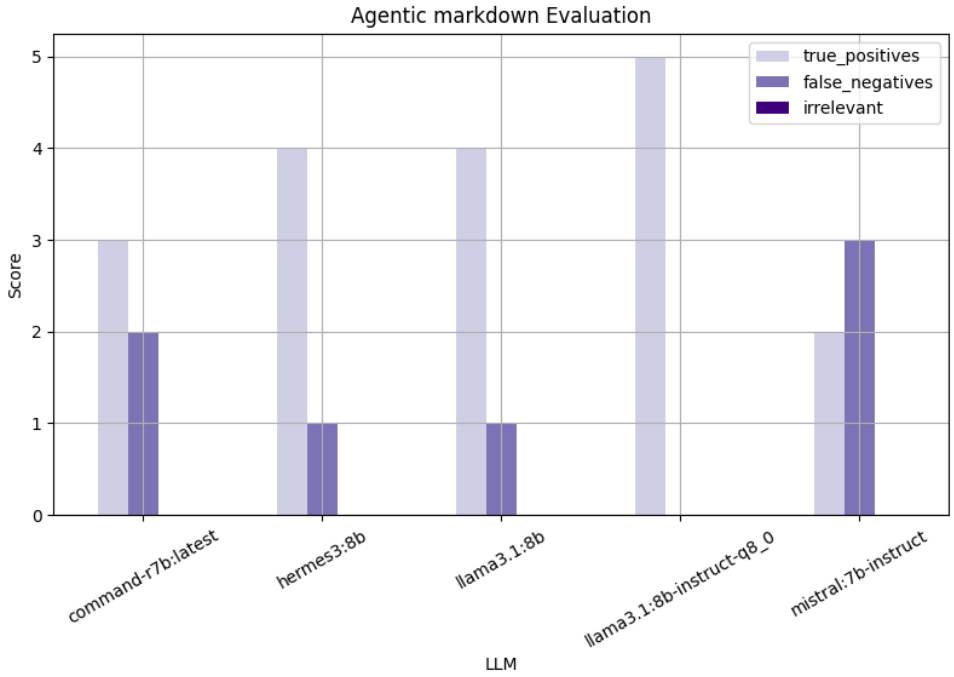

Figure 3: Evaluation outcome counts for setup C.

Figure 4 shows the average execution time per document for Setup A

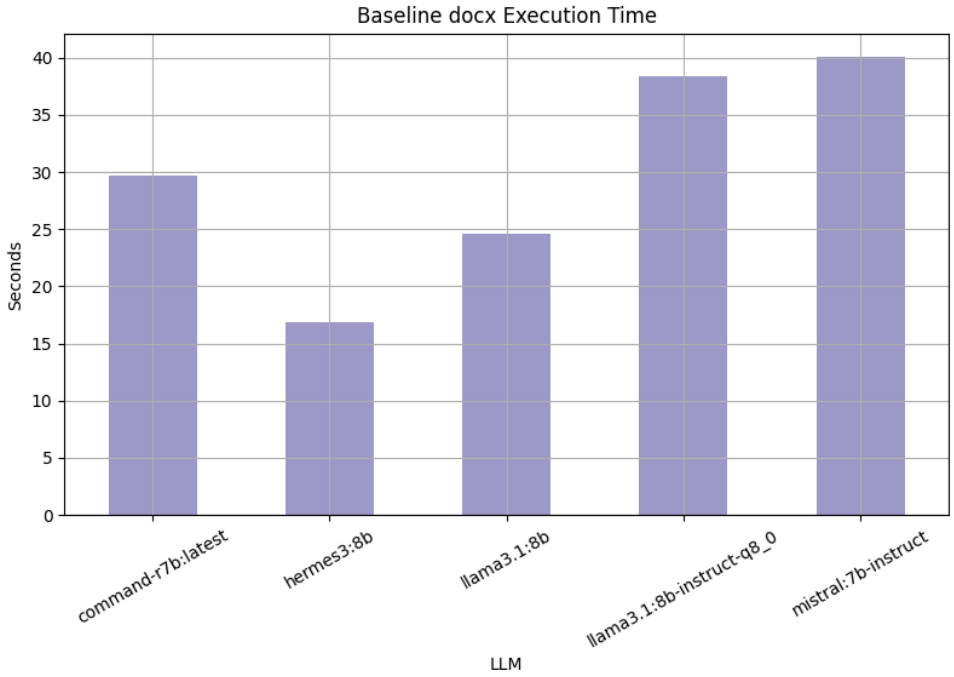

Figure 4: Execution time for setup A

Figure 5 shows the average execution time per document for Setup B

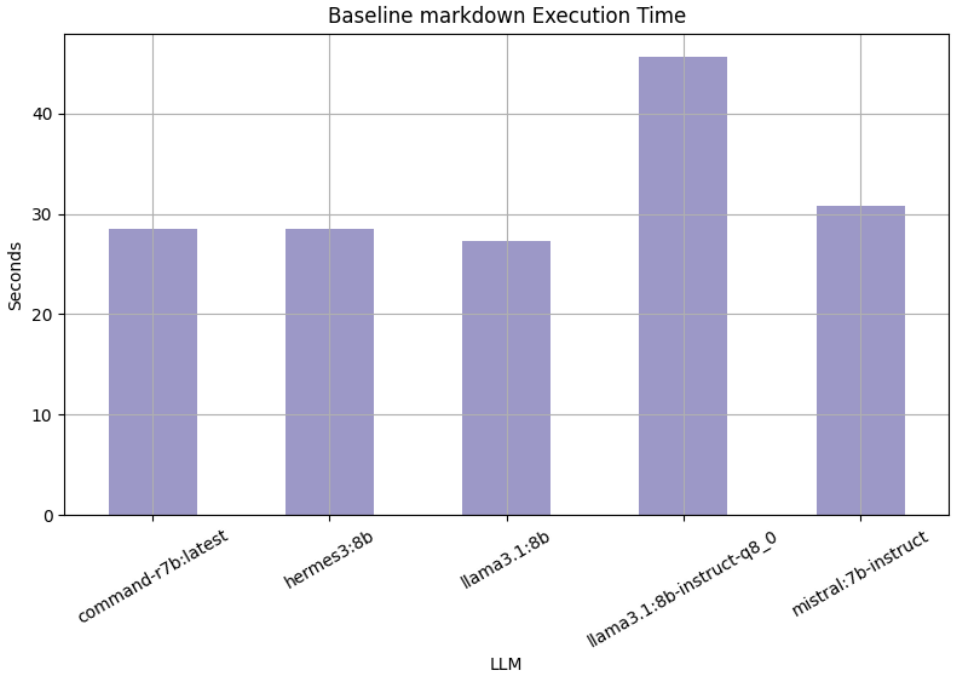

Figure 5: Execution time for setup B

Figure 6 shows the average execution time per document for Setup C

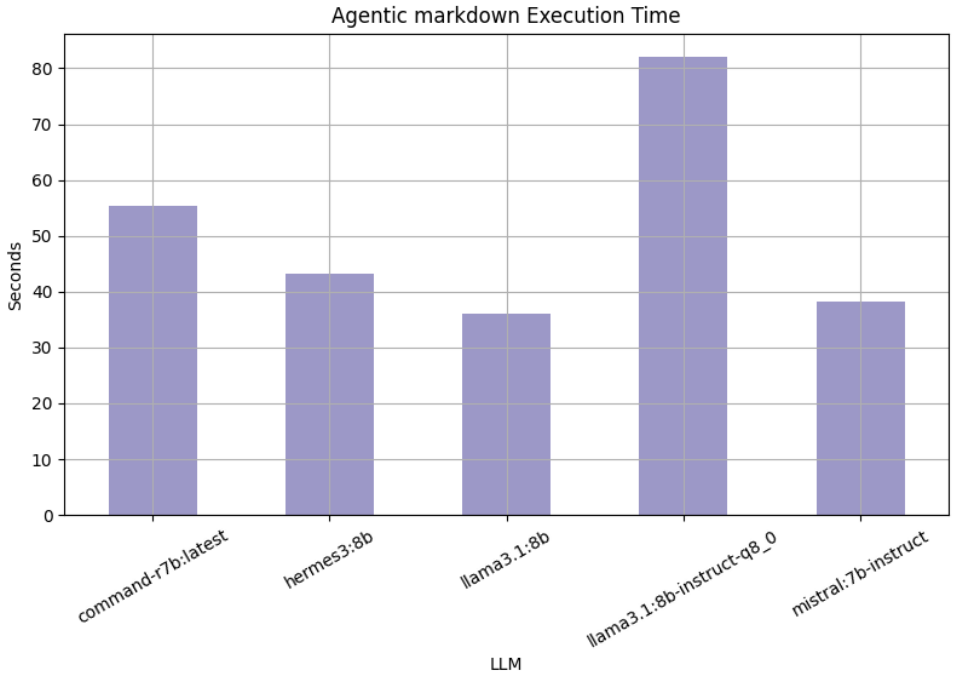

Figure 6: Execution time for setup C

5 Discussion

The results highlight a major challenge in the preprocessing pipeline: the correct formatting of input documents. When using `python-docx`, footers are not reliably removed, and paragraph boundaries are inconsistently extracted. This significantly impacts model performance—inputs derived directly from DOCX files led to noticeably worse results compared to manually formatted markdown documents. These findings underscore the critical importance of structured, well-formed input when working with language models in clinical document analysis. Further experiments revealed that even minor variations in the system prompt—such as the presence or absence of a comma or blank line—can alter the model's output. This sensitivity suggests that prompt design and input consistency must be treated as part of the optimization process.[2] Additionally, we explored different prompting strategies, including zero-shot, one-shot, and few-shot configurations. In zero-shot prompting, the model receives only task instructions; in one-shot or few-shot prompting, it also receives one or more examples to guide its behavior.[3] Surprisingly, in our experiments, one-shot and few-shot examples consistently degraded performance: models often reproduced the given example as their output rather than applying the underlying logic to the input. This suggests that small quantized models are highly sensitive to surface-level patterns and may lack the abstraction capacity required for effective in-context generalization in complex medical tasks.

The execution time increases as expected when using agentic workflows, introducing a trade-off between speed and evaluation accuracy. While this overhead is justified by improved performance in complex reasoning tasks, it remains a limiting factor in time-sensitive applications. The expanded context window enables the system to re-inject previous responses and refine evaluations iteratively, which is particularly useful in multi-step assessments. However, if the evaluation can be performed in a single turn without follow-up interactions, the context length can be reduced (e.g., to 8192 tokens) to conserve computational resources without significantly degrading performance.

Another complicating factor is the lack of a universally accepted definition of what constitutes a "complete" medical discharge report. Standards may vary not only across countries but even between institutions, making generalization difficult. Moreover, human based evaluation of LLMs in general can be problematic due to the inherent subjectivity of human judgement.[4]

In this work, the agentic evaluation setup focused exclusively on the diagnosis section, as this was not reliably detected as incomplete in the baseline setup—even when using a properly formatted document. In principle, the same methodology could be applied to other sections, though this would further increase computational cost and latency.

One potential direction for future work is the integration of a retrieval-augmented generation (RAG) component. This would allow individual clinics to provide their own reporting conventions and medical standards, enabling more context-aware evaluations.[5] Combined with fine-tuning on high-quality institution-specific data, this approach could not only improve section completeness detection but also allow the model to adapt its tone and terminology to local clinical norms.

6 Conclusion

This thesis investigated whether small, open-source language models can be optimized to evaluate the completeness of medical discharge reports in a manner comparable to proprietary systems such as ChatGPT. The results show that such models—particularly those in the 7–8B parameter range—can be effectively adapted to perform section-level completeness evaluation, especially when guided by carefully designed prompts and lightweight agentic workflows.

However, this performance was demonstrated under a simplified evaluation framework based on binary classification (complete vs. incomplete) for individual report sections. While the approach enables clear metric comparison and model alignment, it does not yet capture the full nuance of clinical document quality, such as partial completeness, semantic adequacy, or medical relevance.

Future work should extend this framework to more complex classification schemes and richer annotation layers, enabling models not only to detect missing content but also to assess clinical plausibility, redundancy, or contradiction. Nonetheless, the findings presented here highlight the potential of small, local LLMs in supporting structured clinical documentation analysis—particularly in privacy-sensitive or resource-constrained environments.

7 Appendix

- Project repository on GitHub

References

[1] "Large language models can support generation of standardized discharge summaries – A retrospective study utilizing ChatGPT-4 and electronic health records". In: *International Journal of Medical Informatics* (2024).

[2] "Does Prompt Formatting Have Any Impact on LLM Performance?" In: *arXiv:2411.10541v1* (2024).

[3] "How to Prompt? Opportunities and Challenges of Zero- and Few-Shot Learning for Human-AI Interaction in Creative Applications of Generative Models". In: *arXiv:2209.01390v1* (2022).

[4] "Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences". In: *Association for Computing Machinery* (2024).

[5] "Medical Graph RAG: Towards Safe Medical Large Language Model via Graph Retrieval-Augmented Generation". In: *arXiv:2408.04187v2* (2024).