Optimal LLM Size for Medical Document Classification Using Context Engineering

Data Sovereignty Procedures for Doctors (DSP4D)

Semesterarbeit

Studiengang:	CAS Generative KI
Autor*in:	Benjamin Haegler, Christian Sprecher
Betreuer*in:	[Betreuer einfügen]
Auftraggeber*in:	[Auftraggeber einfügen]
Expert*in:	[Experte einfügen]
Datum:	2025

Berner
Fachhochschule

Abstract

This paper investigates the minimum viable Large Language Model (LLM) size required for reliable medical document classification and clinical action generation. We evaluate multiple context engineering strategies—including few-shot learning, retrieval-augmented generation (RAG), and long-context approaches—to determine optimal trade-offs between model size, inference cost, and clinical accuracy. Our experiments focus on edge deployment scenarios where data sovereignty requirements mandate local processing.

Keywords: Large Language Models, Few-Shot Learning, Medical Document Classification, Edge Deployment, Data Sovereignty

Inhaltsverzeichnis

1 Introduction

The healthcare sector is currently operating under substantial strain, compelled to enhance operational efficiency while simultaneously upholding rigorous standards of data privacy and patient safety. The workload borne by general practitioners (GPs) has intensified markedly due to the proliferation of administrative responsibilities. Following direct patient consultations, practitioners frequently dedicate hours to the scrutiny and triage of incoming documentation—ranging from laboratory reports and referrals to insurance correspondence—as well as the drafting of replies and the maintenance of patient records. This administrative burden results in significant latency and cognitive fatigue, typically accumulating during the period subsequent to clinic closure.

The core strategic challenge, therefore, lies in automating these documentation and correspondence workflows to alleviate physician workload, without compromising Data Sovereignty Procedures. Conventional cloud-based solutions present considerable regulatory complexity or are explicitly prohibited in many jurisdictions due to the acute sensitivity of medical data. Consequently, there is an explicit requirement to engineer solutions that necessitate neither reliance on external online services nor the integration of prohibitively expensive hardware infrastructure.

The emergent technical opportunity to resolve this dichotomy is found within Generative Artificial Intelligence (GenAI). Through the deployment of locally hosted Large Language Models (LLMs), it becomes feasible to deliver high-performance AI functionality in a decentralised manner, entirely severed from external server connectivity. This architecture facilitates the strict implementation of Data Sovereignty Procedures directly on the physician's local workstation. This project, therefore, addresses the critical imperative to identify a resource-efficient paradigm that enables the viable deployment of GenAI on standard local hardware, bridging the gap between advanced automation and strict data governance.

1.1 Motivation

The operational reality of modern general practice is increasingly characterised by a disproportionate imbalance between clinical patient care and administrative overhead. Post-consultation hours are frequently dominated by the cognitive burden of reviewing complex medical documentation and generating necessary correspondence. This systemic inefficiency does not merely represent a temporal inconvenience; it contributes significantly to physician burnout and reduces the net time available for patient interaction. Consequently, there is an urgent imperative to deploy automated systems capable of absorbing this clerical workload. However, the integration of such automation creates a complex technological dilemma regarding the ethical and legal frameworks governing medical confidentiality.

The fundamental problem inhibiting the widespread adoption of Generative AI in this domain is the architectural reliance of current State-of-the-Art (SOTA) solutions on cloud infrastructure. While commercial Large Language Models (LLMs) offer the requisite reasoning capabilities to triage and summarise medical data, their deployment typically necessitates the transmission of sensitive Patient Health Information (PHI) to third-party servers. This architecture presents an unacceptable risk profile regarding Data Sovereignty Procedures. In many jurisdictions, sending unredacted medical records to external API endpoints violates strict data protection regulations. Thus, practitioners face a dichotomy: utilise powerful cloud-based tools at the risk of regulatory non-compliance, or forego AI assistance entirely. There is a distinct lack of validated frameworks that enable the deployment of effective, high-quality AI models within the secure environment of a local practice without necessitating prohibitively expensive enterprise-grade hardware.

Addressing this technological and regulatory gap, this thesis centres on the critical question of how an algorithmic selection framework can be developed and validated to identify the most resource-efficient Large Language Model (LLM) capable of operating locally. The objective is to relieve physicians of documentation and correspondence tasks while strictly maintaining data sovereignty. This inquiry implies the necessity of establishing a balance between computational efficiency—specifically regarding inference speed and memory footprint—and semantic accuracy, ensuring that the shift to decentralised processing does not result in a degradation of output reliability.

1.2 Research Questions

1. What is the minimum model size for reliable document classification (>95% accuracy)?
2. How do different context engineering strategies affect the size-accuracy trade-off?
3. Can sub-3B parameter models achieve clinical safety standards with appropriate context?

2 Theory / State of Research

Evaluating the performance of language models requires quantifiable metrics that capture both accuracy and semantic quality. While subjective assessment remains valuable, reproducible benchmarks enable systematic comparison across models and configurations. This section reviews established evaluation frameworks — from classical NLP metrics through modern LLM-based assessment methods — and situates them within the medical domain where accuracy requirements are particularly stringent.

2.1 Evaluations in Classical Text Analysis

In classical natural language processing (NLP) and information retrieval, evaluation relies heavily on comparing system output against a “gold standard” or ground truth. These metrics are particularly relevant for classification tasks, such as identifying clinical intent or extracting specific medical entities.

2.1.1 String Similarity & Edit Distance

When exact matches are too strict, string similarity metrics quantify the difference between two sequences.

- **Levenshtein Distance** (or Edit Distance) counts the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word or text string into the other (Levenshtein 1966). This is valuable for correcting typos or measuring near-matches in entity extraction.

2.1.2 Classification Metrics

For tasks involving categorization, the confusion matrix serves as the foundation for most metrics, tracking true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) (Manning, Raghavan, und Schütze 2008).

- **Accuracy** measures the overall correctness of the model but can be misleading in unbalanced datasets, which are common in medical contexts (e.g., rare diseases).
- **Precision** (Positive Predictive Value) measures the proportion of identified positive cases that were actually correct. In a clinical setting, high precision minimizes false alarms.
- **Recall** (Sensitivity) measures the proportion of actual positive cases that were identified. High recall is critical in medicine to ensure no pathology is overlooked.
- **F1-Score** provides the harmonic mean of precision and recall, offering a balanced view when finding a compromise is necessary (Sokolova und Lapalme 2009).

2.1.3 Generation Metrics

For tasks involving text generation, such as summarizing findings or suggesting actions, classical n-gram based metrics are often employed:

- **BLEU (Bilingual Evaluation Understudy)** measures the precision of n-grams in the generated text compared to reference texts. While popular, it is often criticized for focusing only on exact matches and ignoring semantic meaning (Papineni u. a. 2002).
- **METEOR (Metric for Evaluation of Translation with Explicit ORdering)** improves upon BLEU by incorporating stemming and synonym matching, resulting in better correlation with human judgment (Banerjee und Lavie 2005).
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** focuses on recall, measuring how much of the reference text appears in the generated output, widely used for summarization (Lin 2004).

While these metrics provide objective, reproducible scores, they often correlate poorly with human judgment for complex reasoning tasks, necessitating more advanced evaluation paradigms.

2.1.4 Semantic & Embedding-based Metrics

To overcome the limitations of exact n-gram matching, semantic metrics utilize word or sentence embeddings to measure similarity in meaning rather than just surface form.

- **BERTScore** computes a similarity score for each token in the candidate sentence with each token in the reference sentence using contextual embeddings (e.g., from BERT). This allows for a more robust evaluation of paraphrases and synonyms (Zhang u. a. 2020).
- **Word Mover's Distance (WMD)** and its variants (like MoverScore) measure the minimum "distance" required to move the embedded words of one document to the other. This approach captures semantic distance effectively, even when no words overlap (Kusner u. a. 2015; Zhao u. a. 2019).

2.1.5 LLM-Based Evaluation (LLM-as-a-Judge)

Recent advances have shifted towards using Large Language Models themselves as evaluators, a paradigm known as "LLM-as-a-Judge". This approach uses the reasoning capabilities of capable models (such as GPT-5) to assess the quality of generated text based on complex criteria such as helpfulness, safety, and coherence, often achieving higher correlation with human judgment than traditional metrics.

- **G-Eval** is a framework that uses LLMs with Chain-of-Thought (CoT) reasoning to evaluate generated text. By decomposing the evaluation task into a series of steps, it provides fine-grained scores that align closely with human preference (Liu u. a. 2023).

- **GPTScore** evaluates texts by calculating the probability of the generated text given a specific instruction or context, using the model's own likelihood scores as a proxy for quality (Fu u. a. 2024).
- **Prometheus** is an open-source LLM specifically fine-tuned for evaluation purposes. It allows for custom evaluation criteria and feedback generation, offering a cost-effective alternative to using proprietary models like GPT-4 as judges (Kim u. a. 2024).
- **Ragas** (Retrieval Augmented Generation Assessment) is a framework specifically designed for evaluating RAG pipelines. It defines metrics such as *context precision*, *faithfulness*, and *answer relevancy*, using an LLM to verify if the generated answer is grounded in the retrieved documents and if it actually answers the user's question (Es u. a. 2024).

TODO CHS: DAG ### Evaluation Challenges

Despite the proliferation of evaluation frameworks, assessing LLM quality remains a central limitation in the field. The metrics described above each carry inherent weaknesses that complicate reproducible benchmarking.

Weakness of Traditional Metrics. Automated measures such as BLEU and ROUGE correlate only weakly with human judgment in many contexts (Reiter 2018). These metrics rely on n-gram overlap and fail to capture semantic equivalence, coherence, or reasoning quality. A generated response may convey the correct meaning through paraphrasing yet receive a low score due to lexical divergence from the reference text. Conversely, a response with high word overlap may be factually incorrect or incoherent. This limitation is particularly acute in medical contexts, where semantic accuracy matters more than surface-level similarity.

Bias in LLM-as-a-Judge. While LLM-based evaluation addresses some limitations of traditional metrics, it introduces new biases. Research has identified a *self-preference bias*: models systematically favor outputs generated by themselves or similar architectures over those from other models (Panickssery, Bowman, und Feng 2024). Additionally, a *length bias* causes LLM judges to prefer longer responses regardless of quality, conflating verbosity with helpfulness (Saito u. a. 2024). These biases undermine the reliability of automated evaluation pipelines and complicate cross-model comparisons.

Data Contamination. Many established benchmarks (MMLU, HellaSwag, GSM8K) are publicly available on the internet, raising the risk that models have encountered test items during pretraining (Sainz u. a. 2024). When benchmark data appears in training corpora, evaluation scores become inflated and no longer reflect genuine generalization capability. This contamination problem is difficult to detect and increasingly prevalent as training datasets grow to encompass ever-larger portions of the web. For medical applications, this raises questions about whether reported performance on clinical benchmarks reflects true capability or mere memorization.

These challenges underscore the need for multi-faceted evaluation approaches that combine automated metrics with human assessment, use held-out test sets, and interpret results with appropriate caution.

For the present study, these limitations are partially mitigated by our reliance on relative rather than absolute metric comparisons; nevertheless, they remain relevant considerations when interpreting results.

2.2 LLM in the Context of Medical Science

The application of Large Language Models (LLMs) in medicine is an evolution of clinical Natural Language Processing (NLP), which gained significant momentum with the release of specialized models like ClinicalBERT (Alsentzer u. a. 2019). While early models focused on entity recognition and extraction, modern LLMs offer the potential to summarize charts and suggest clinical actions. However, their integration into clinical workflows is constrained by critical requirements for accuracy, data privacy, and data sovereignty.

2.2.1 Privacy, Security, and Data Sovereignty

The use of cloud-based LLMs in healthcare introduces significant risks that have been documented since the early days of transformer models.

- **Data Leakage and Memorization:** Foundational research has shown that LLMs can memorize and inadvertently "regurgitate" sensitive training data, including personally identifiable information (PII) (Carlini u. a. 2021). In a medical context, this poses a risk of exposing protected health information (PHI) through model outputs.
- **Adversarial Vulnerabilities:** Modern aligned models are susceptible to adversarial attacks, such as prompt injection, which can bypass safety filters and potentially lead to the disclosure of sensitive context or the generation of incorrect medical advice (Zou u. a. 2023).
- **Ethical and Regulatory Gaps:** A 2025 scoping review identifies a persistent lack of ethical oversight and informed consent in many LLM-based medical studies, highlighting an urgent need for privacy-preserving architectures (Zhong u. a. 2025).

To mitigate these risks, researchers are exploring **Data Sovereignty**—the principle that health data should remain under the control of the originating institution or the patient. This has led to two main research directions:

1. **On-Device Deployment:** Operating models entirely on local hardware (e.g., Jetson Nano) to ensure no sensitive data ever leaves the clinical environment (Wu u. a. 2025).
2. **Privacy-Preserving Training:** Techniques like “Whispered Tuning” and differential privacy are being developed to prevent PII memorization during model adaptation (Singh u. a. 2024).

2.2.2 Specialized Medical Applications

Dual-stage and Lightweight Patient Chart Summarization
Wu et al. (2025) proposed a dual-stage system specifically for emergency departments. By using a Small Language Model (SLM) on embedded devices, they demonstrate that it is possible to provide actionable clinical summaries without cloud dependencies, thereby fulfilling the highest standards of data sovereignty (Wu u. a. 2025).

ELMTEX: Structured Clinical Information Extraction
Guluzade et al. (2024) showed that fine-tuned smaller models can outperform larger, general-purpose counterparts in extracting structured data from unstructured German clinical reports. Their work demonstrates that for specialized medical tasks, increased parameter count does not guarantee improved performance — a finding that supports the feasibility of local deployment (Guluzade u. a. 2025).

GraSCCo: A Foundation for Privacy-Preserving Research
The Graz Synthetic Clinical text Corpus (GraSCCo) remains a cornerstone for this research area. As a multiply-alienated German clinical corpus, it allows researchers to benchmark models on realistic medical narratives without the legal and ethical risks associated with real patient data (Modersohn u. a. 2022; Lohr u. a. 2025).

2.3 Scaling Laws and Model Efficiency

A central question for deploying LLMs in privacy-sensitive environments is: how small can a model be while maintaining acceptable performance? Early scaling laws suggested a straightforward trade-off, but recent developments in Small Language Models (SLMs) have significantly shifted expectations.

2.3.1 Historical Context

Early work by Kaplan et al. (2020) and Hoffmann et al. (2022) established that language model performance follows predictable power-law relationships with model size and training data

(Kaplan u. a. 2020; Hoffmann u. a. 2022). While foundational, these findings predate the current generation of highly optimized small models and do not fully capture the capabilities of modern SLMs.

2.3.2 The Rise of Small Language Models

A comprehensive survey by Lu et al. (2024) benchmarked 59 SLMs (100M–5B parameters) across commonsense reasoning, mathematics, and in-context learning tasks. Their findings reveal substantial performance improvements: SLMs improved by 10–13% between 2022 and 2024, outpacing larger models which improved by only 7.5% over the same period (Lu u. a. 2024). Notably, the Phi-3 model (3.8B parameters) achieves 69% on MMLU — performance comparable to Mixtral 8x7B and GPT-3.5. This demonstrates that modern SLMs, through optimized architectures and high-quality training data, can compete with models several times their size.

2.3.3 A Note on Terminology

The term “Small Language Model” warrants clarification. In current usage, “small” refers exclusively to parameter count — not to training data scope. A 3B parameter model trained on trillions of web-scale tokens is considered “small” only relative to 70B+ frontier models. This stands in contrast to *domain-specific* models such as ClinicalBERT or PubMedBERT, which are smaller in both parameters and training scope, having been trained on specialized medical corpora. Throughout this thesis, the term SLM refers to language models with fewer than 100 billion parameters, regardless of their training data origin. This broader definition encompasses both general-purpose compact models (Phi, Qwen, Llama) and domain-specialized models, allowing for comparison across deployment scenarios.

2.3.4 Capability Density and the Densing Law

Xiao et al. (2025) formalize this trend through the concept of *capability density* — defined as capability per parameter. Their empirical analysis reveals a “densing law”: capability density approximately doubles every 3.5 months (Xiao u. a. 2024). This trajectory indicates that equivalent performance can be achieved with exponentially fewer parameters over time, making local deployment increasingly viable.

2.3.5 Edge Deployment Considerations

Recent work specifically addresses SLM deployment on resource-constrained devices. Hassanpour et al. (2025) systematically evaluate SLMs for edge scenarios, examining the trade-offs between model size, quantization levels, and task performance (Lu u. a. 2025). Their findings confirm that sub-4B parameter models can achieve practical utility for domain-specific tasks when properly configured — a key consideration for medical applications where data must remain on-device.

2.3.6 Implications for This Study

These developments frame the research question: given hardware constraints of on-device deployment for sensitive medical data, what is the smallest pre-trained model that can reliably perform clinical document classification? The answer depends not only on parameter count, but also on model generation and — as the following section explores — context engineering strategies that can augment smaller models at inference time.

2.4 Context Engineering Strategies

TBD what else do we discuss here?

2.4.1 Comprehensive Comparison of Prompting Techniques

This table evaluates techniques based on their ability to extract accurate, structured “Ground Truth” (Golden Answers) from the GraSCCo medical corpus. The comparison of techniques is equally relevant for prompting the set of smaller LLMs in the evaluation phase.

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
Zero-Shot Prompting	Asking the model to perform a task without examples.	“Extract all diagnoses from this text.”	Fast and low token cost. Useful for checking the baseline capability of a model.	High risk of hallucination and format inconsistency. The model may guess the required medical style incorrectly.	Prompt Engineering GuidePrompt engineering techniques: Top 6 for 2026
Few-Shot Prompting	Providing examples (input-output pairs) within the prompt.	“Style Guide: You provide 3 examples of GraSCCo raw text and the corresponding perfect”Golden Answer” format.(possible after first supervision session)”	Essential for enforcing the specific syntax and brevity required for the Golden Answers.	The model may overfit to the examples and ignore the nuance of the new input.	Language Models are Few-Shot Learners (NeurIPS)Prompt Engineering GuidePrompt engineering techniques: Top 6 for 2026
Chain-of-Thought (CoT)	Instructing the model to generate intermediate reasoning steps.	Clinical Reasoning: “First, list all medications found. Second, check if they are current or historical. Finally, output the list.”	Critical for connecting implied symptoms to explicit medical codes. Reduces “skipping” of details.	Increases token usage. Requires parsing to separate the “thought” from the “golden answer.”	Chain of Thought Prompting Elicits reasoning (arXiv)Prompt Engineering GuidePrompt engineering techniques: Top 6 for 2026

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
Self-Consistency	Generating multiple outputs for the same prompt and selecting the most frequent one.	Validation: Generate the summary 5 times. If “Diabetes Type 2” appears in 5/5, keep it. If “Hypertension” appears in 1/5, discard it.	The best statistical defense against hallucinations. Essential for creating a robust “Gold Standard”.	Computationally expensive (requires N times the inference cost).	Self-Consistency Improves Chain of Thought (arXiv)Prompt engineering techniques: Top 6 for 2026
Skeleton-of-Thought (SoT)	Generating a skeleton/outline first, then expanding points in parallel.	Structure Planning: 1. Generate list of headers (Dx, Rx). 2. Fill sections in parallel.	Accelerates generation speed (up to 2.39x). Good for long, structured discharge summaries.	Suited for writing new content, less proven for extracting specific facts from existing chaos.	Skeleton-of-Thought: Prompting LLMs for Efficient Parallel Generation[ICLR 2024] Skeleton-of-Thought: Prompting LLMs for Efficient Parallel Generation (GitHub)
Prompt Chaining	Breaking a task into subtasks where output A becomes input B.	Workflow: 1. Extraction Prompt -> 2. Filtering Prompt -> 3. Formatting Prompt.	High reliability. Isolates errors. Allows for intermediate transformation (e.g., cleaning citations).	Requires building a controller application (state management) between prompts.	Prompt Engineering Guide: Prompt Chaining (GitHub)PromptChainer Paper (arXiv)

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
Role / Persona Prompting	Assigning a specific role/profession to the AI.	Context Setting: “You are a Senior Chief Physician at a Swiss hospital…”	Drastically improves tone and handling of medical abbreviations/jargon.	Can lead to verbosity if the persona is too “chatty.”	Prompt Engineering: Part 2 - Best Practices for Software Developers in Digital IndustriesPrompt Engineering GuidePrompt engineering techniques: Top 6 for 2026
Multi-Persona Prompting	Simulating a discussion between multiple agents (e.g., Drafter & Reviewer).	Quality Assurance: Agent A extracts data; Agent B reviews it for missing info; Agent C finalizes.	Simulates a “four-eyes principle” (peer review), reducing errors through internal debate.	High latency and token cost; complex to orchestrate.	Exploring Multi-Persona Prompting for Better Outputs

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
Tree of Thoughts (ToT)	Exploring multiple reasoning paths and back-tracking.	Complex Triage: Exploring different diagnostic possibilities before committing to one.	Useful for complex differential diagnosis problems.	Likely overkill for extraction tasks where the answer is explicitly in the text.	Tree of Thoughts: Deliberate Problem Solving with Large Language Mo-dels[NeurIPS 2023] Tree of Thoughts: Deliberate Problem Solving with Large Language Models
Retrieval Augmen-ted Generati-on (RAG)	Retrieving external data to ground the generati-on.	Fact Checking: Using a vector DB to validate if an extracted drug name exists in RxNorm.	Prevents hallucination of non-existent drugs; ensures terminology standardization.	Requires external database infrastruc-ture. And a document base.	Prompt enginee-ring techni-ques: Top 6 for 2026Prompt Enginee-ring Guide: Prompt Chaining (GitHub)
Automatic Prompt Engineer (APE)	Using an LLM to generate and optimize prompts.	Asking GPT-4 to write the optimal prompt for analyzing GraSCCo texts.	Saves time on trial-and-error.	The resulting prompt might be obscu-re/hard to interpret.	Prompt Enginee-ring Guide

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
Generated Knowledge Prompting	Asking the model to generate relevant knowledge before answering.	“List common side effects of Ibuprofen, then summarize the patient’s complaints regarding medication.”	Can help if the medical text is ambiguous (e.g., vague symptoms), providing context for the summary.	Risk of generating false knowledge (hallucinated medical facts) which then contaminates the summary.	Prompt Engineering Guide
Automatic Reasoning and Tool-use	Allowing the LLM to use external tools (calculators, APIs).	Calculating cumulative dosage or converting units (e.g., mg to g) found in the text.	Ensures mathematical accuracy in the medical record (e.g., total radiation dose).	Adds complexity; the model might fail to invoke the tool correctly.	Toolformer: Language Models Can Teach Themselves to Use Tools (arXiv)
Active-Prompt	Selecting the most uncertain examples for human annotation to teach the model.	Identifying GraSCCo texts where the model is “unsure” and asking a doctor to manually create the Golden Answer.	Maximizes the value of human expert time (efficient annotation).	Requires a human-in-the-loop workflow.	Prompt Engineering Guide
Directional Stimulus Prompting	Using a separate small model to generate “hints” or keywords to guide the main LLM.	Extracting keywords (e.g., “Heart”, “Attack”) first, then feeding them to the LLM to generate the summary.	Can focus the model on specific medical sections (e.g., “Focus only on cardiac events”).	Requires training or prompting an auxiliary policy model.	Prompt Engineering Guide
Program-Aided Language Models (PAL)	Generating code to solve reasoning steps.	Writing a Python script to extract and sort dates of admission from the text.	Extremely precise for structured data extraction (dates, dosages).	Fails if the medical text is too unstructured or uses ambiguous natural language.	Prompt Engineering Guide

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
ReAct (Reasoning + Acting)	Interleaving reasoning traces with action execution.	“Thought: I need to check if this drug interacts with... Action: Search drug database.”	Good for clinical decision support agents.	Overly complex for the specific task of generating static Golden Answers from text.	REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS (arXiv)Prompt Engineering Guide: ReAct Prompting (GitHub)
Reflexion	An agent reflecting on past mistakes to improve future responses.	The model generates a summary, checks it against rules, critiques itself (“I missed the date”), and rewrites.	Improves quality iteratively without human intervention.	Can get stuck in loops if the self-critique is flawed.	Reflexion: Language Agents with Verbal Reinforcement LearningPrompt Engineering: Part 2 - Best Practices for Software Developers in Digital Industries
Multimodal CoT	Chain-of-Thought with images and text.	Analyzing X-rays alongside the radiology report.	Essential if the GraSCCo corpus contained images (it does not, it is text-based).	Not applicable to text-only medical corpora.	Prompt Engineering Guide

Technique	Description	Application to Medical Golden Answers	Pros for Medical Records	Cons / Risks	References
Graph-Prompting	Representing data as a graph structure within the prompt.	Mapping patient symptoms to a knowledge graph of diseases.	Good for understanding relationships (Symptom A -> Disease B).	Text-to-Graph conversion is difficult and error-prone.	Graph of Thoughts: Solving Elaborate Problems with Large Language Models
Meta-Prompting	Asking the model to assume a persona or higher-level view.	"Act as a senior medical consultant reviewing a junior doctor's note."	Can improve the tone and professionality of the output.	Mostly affects style, less impact on factual extraction accuracy.	Prompt engineering techniques: Top 6 for 2026Prompt Engineering Guide

3 Methodology

Development of an Algorithmic Framework for Resource-Efficient Local LLM Selection

The primary objective of this study is the development of an algorithmic selection framework designed to identify the most resource-efficient Large Language Model (LLM) suitable for local execution. By validating output quality against a set of verified "Golden Answers", this research seeks to establish an optimal equilibrium between computational performance and data sovereignty. The proposed algorithm argues for a shift away from maximalist parameter counts towards targeted efficiency without compromising output fidelity.

3.1 Procedure

The research design follows a rigorous four-phase methodological approach to ensure reproducibility and statistical significance:

3.1.1 Phase I: Dataset Curation and Establishment of Ground Truth

The initial phase focuses on the identification and preprocessing of a stable text corpus. This corpus serves as the foundational bedrock for deriving "Golden Answers" (Ground Truth). Esta-

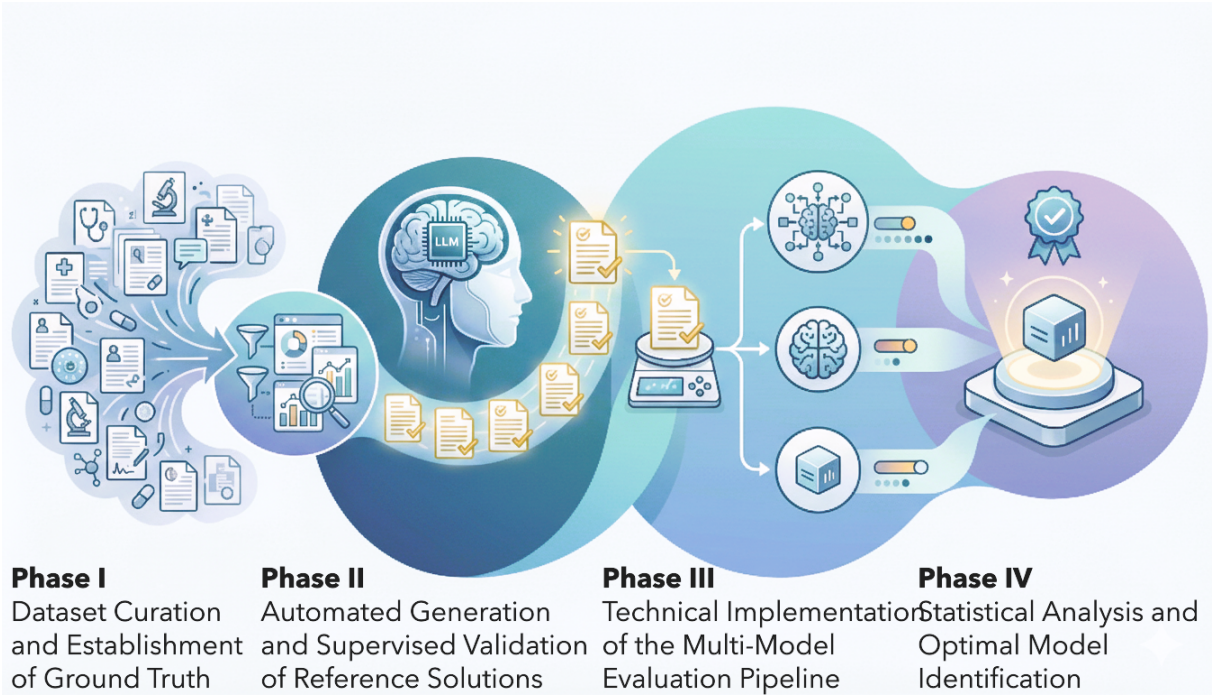


Abbildung 1: four-phase methodological approach

blishing this baseline is critical, as it functions not only for the initial assessment of the chosen State-of-the-Art (SOTA) LLM but also acts as the immutable comparative benchmark during the subsequent model evaluation phases.

3.1.2 Phase II: Automated Generation and Supervised Validation of Reference Solutions

In this step, a selected high-performance LLM is utilised to generate high-fidelity “Golden Answers”. To ensure domain-specific accuracy, these outputs undergo a supervised review and validation process by a qualified subject matter expert (General Practitioner). Concurrently, various prompt engineering techniques are evaluated, with sessions systematically logged. This data retention is essential to argue whether complex prompting strategies yield comparable performance enhancements when applied to significantly smaller models later in the process.

3.1.3 Phase III: Technical Implementation of the Multi-Model Evaluation Pipeline

A robust evaluation framework is engineered to assess a diverse array of LLMs, varying in architecture, quantisation, and parameter size. The system is designed to task these models with reproducing the “Golden Answers” derived from the corpus in Phase I. Consistent with Phase II, the previously identified prompt engineering strategies are re-evaluated within this constrained environment. The pipeline captures comprehensive performance metrics, generating the

necessary empirical data input for the final analysis.

3.1.4 Phase IV: Statistical Analysis and Optimal Model Identification

The concluding phase involves a multi-dimensional assessment of the generated data to isolate the optimal model. This includes the application of context-aware content metrics as well as an “LLM-as-a-Judge” paradigm to comparatively evaluate the semantic quality of the outputs. By synthesising these qualitative and quantitative insights, the study identifies the specific LLM that strictly adheres to the pre-defined requirements, thereby validating the feasibility of high-quality, local, and resource-efficient generative AI.

3.2 Data Source: GraSCCo

Instead of generic document types, this research utilizes the **Graz Synthetic Clinical text Corpus (GraSCCo)** (Lohr u. a. 2025; Modersohn u. a. 2022).

GraSCCo is the first publicly shareable, multiply-alienated German clinical text corpus, designed specifically for clinical NLP tasks without compromising patient privacy.

The corpus provides a diverse set of clinical scenarios, which we use to evaluate the models’ ability to classify document intent and generate appropriate clinical actions based on German-language clinical reports.

The task we give the models is to update a patients health record (HBA) based on supplied clinical report.

3.3 Golden Answer Generation

Due to the lack of a specialized medical background, we use a State-of-the-Art (SOTA) Large Language Model (LLM) to generate initial “Silver Answers”. These serve as preliminary structured outputs derived from the GraSCCo medical corpus. To ensure the high quality and clinical validity of these answers, a subset of the LLM-generated responses is undergo evaluation by one or more medical experts. This human-in-the-loop verification allows us to refine the outputs into a “Gold Standard” (Golden Ansers) additionally suitable for benchmarking smaller models.

3.3.1 Preparation Work

To ensure a structured and scientifically sound prompt engineering process, the following preparatory steps were undertaken in collaboration with medical professionals.

Medical Context Stratification The reports from the **GraSSCCO** corpus were categorized into specific medical fields. This stratification allows for a granular comparison of model performance across different clinical contexts and enables an evaluation of the models' ability to correctly assign documents to their respective domains.

The following categories were defined for this study:

- Oncology
- Neurology
- Psychiatry
- Cardiology
- Internal Medicine
- Surgery
- Orthopedics
- Ophthalmology
- Dermatology

Standardized Output Format In collaboration with a **General Practitioner (GP)**, a simplified output format was developed. This structure serves as the template for the prompt's output, allowing for the isolated evaluation of partial results and specific data extraction capabilities.

The standardized format consists of the following six sections:

1. **Categories:** One or more precise categories from the predefined list above.
2. **Date and Source:** The date of the report and the issuing entity (e.g., institute, clinic, or specific physician).
3. **Diagnosis:** The specific diagnosis as documented by the author of the original report.
4. **Relevant Metrics:** Extraction of laboratory values, measurement data (e.g., blood pressure, BMI), and other clinical parameters.
5. **Current or Advised Medications:** A list of medications, specifically distinguishing between the patient's **current** medication and **recommended/prescribed** new treatments.
6. **Follow-up:** Extraction of the next clinical steps or planned interventions mentioned in the report.

Prompt Constraints and Data Integrity To minimize “hallucinations” and ensure clinical reliability, the prompt instructions include strict constraints:

- **Evidence-Based Extraction:** The model is instructed to only output values if there is a clear and unambiguous reference within the source text.
- **Linguistic Consistency:** The output must be generated in the **original language** of the document (German) to maintain technical accuracy and prevent translation errors during the extraction phase.

3.3.2 Selection of Prompting Technique: Chain-of-Thought (CoT)

To generate these Silver Answers, we have selected Chain-of-Thought (CoT) prompting. Based on the Comprehensive Comparison of Prompting Techniques, CoT was chosen over other methods for the following strategic reasons: * Clinical Reasoning Alignment: CoT instructs the model to generate intermediate reasoning steps. In a medical context, this is critical for connecting implied symptoms to explicit medical codes and prevents the model from “skipping” vital clinical details. * Reduced Hallucinations: By breaking down the task—for example, listing medications first, then checking their historical status, and finally formatting the output—the model is less likely to produce the formatting inconsistencies or “guesses” typical of Zero-Shot prompting. * Structural Integrity: Unlike simpler techniques, CoT allows for the separation of the “thought” process from the final “golden answer,” ensuring

See: Comprehensive Comparison of Prompting Techniques

While techniques like Self-Consistency or Multi-Persona Prompting offer higher reliability, they were deemed less efficient for this stage due to significantly higher complexity, computational costs and latency. CoT provides the optimal balance between reasoning depth and token efficiency for clinical document classification.

Feature	Standard Prompt	Chain of Thought Prompt
Processing Style	Pattern matching & Direct Extraction	Logical deduction & Evidence-first
Accuracy	High for simple reports	Superior for complex, conflicting reports
Hallucination Risk	Moderate (may guess missing values)	Lower (reasoning step identifies gaps)
Token Usage	Low (Cost-efficient)	Higher (More verbose output)
Auditability	Difficult (Only the result is visible)	Transparent (You see why it chose a category)

3.3.3 System Prompt: Clinical Data Extraction (CoT)

In a medical context, this is particularly valuable because it forces the LLM to identify the evidence in the text before committing to a category or a medication status. This reduces “lazy” extractions where a model might miss a nuance (like a medication being discontinued).

Used prompt:

```
Role: You are an expert Medical Registrar. Extract data into a structured JSON format.

Constraints:
1. Factuality: Extract information ONLY if explicitly stated.
2. Language: Content values must be in the original document language (German).
3. Format: Output ONLY a single valid JSON object.

Available Categories: You MUST choose one or more from this specific list:
["Onkologie", "Neurologie", "Psychiatrie", "Kardiologie", "Innere Medizin", "Chirurgie", "Ort

Methodology: Use the "internal_monologue" to analyze the text step-by-step before populating

Output Schema:
{
 "internal_monologue": {
  "1": "Identify the documents creation date and author or institutions",
  "2": "List diagnoses and primary reason",
  "3": "Locate numerical metrics",
  "4": "Distinguish current vs. advised medication",
  "5": "Identify follow-up instructions"
 },
 "structured_health_record": {
  "categories": ["Must be from the list above"],
  "date_and_source": "YYYY-MM-DD; Institution/Doctor",
  "diagnosis": "Documented diagnosis",
  "relevant_metrics": "Lab values and vitals",
  "medications": {
      "current": "What the patient is already taking",
      "advised": "New prescriptions or changes"
  },
  "follow_up": "Next steps"
```

```
 }
}

Source Text:
{document}
```

Gold Standard Example (CoT Approach)

3.3.4 Ground Truth Generation and Annotation Platform

To facilitate the seamless generation and validation of these answers, we developed a dedicated web application. This platform serves three primary functions:

- **Accessibility:** It allows researchers and medical experts to access the data and provide feedback from any location at any time.
- **Centralized Storage:** It records both the raw LLM outputs (Silver Answers) and the subsequent expert feedback/corrections.
- **Data Pipeline Integration:** The application is designed to automatically export these validated results into the specific input format required by our evaluation framework, ensuring a smooth transition from annotation to model benchmarking.

The platform consits of following Components:

Session Framework

The core of the platform is organized into Sessions. A Session acts as the functional container for processing input documents into “Silver Answers” and managing the subsequent expert annotation process.

Input Documents

This component manages the medical corpora, specifically the GraSCCo raw text files. Users can upload or reference specific documents that require clinical document classification or data extraction.

Configuration & Prompt Engineering

The platform allows for sophisticated prompt management. While it supports single-prompt execution, it is optimized for Prompt Chaining—breaking complex medical tasks into subtasks (e.g., Extraction -> Filtering -> Formatting) to isolate errors and improve reliability. To ensure clinical

accuracy, users can fine-tune the following model parameters: * Temperature: Controls randomness. For medical extraction, a lower range of 0.2–0.5 is recommended to ensure deterministic, consistent, and predictable outputs. * Max Output Tokens: Defines the response length. We recommend 1024–2048 for concise outputs or 4096–8192 for detailed clinical extractions * Top-K Sampling: Limits the model to the K most likely tokens. A setting of 10–40 balances consistency with the flexibility needed for medical terminology. * Top-P (Nucleus Sampling): Selects tokens based on a cumulative probability P. A value of 0.8–0.9 is ideal for maintaining clinical accuracy while allowing for varied medical phrasing.

Execution & Metrics

This module provides real-time visibility into the generation process. It tracks Execution Status and critical performance metrics, including: * Token Consumption: Monitoring input and output volume. * Cost & Quality: Assessing the financial efficiency and the perceived reliability of the "Silver Answers".

Results & Annotation

Once execution is complete, the platform displays the generated answers for each input document. This interface is designed for the human-in-the-loop phase, allowing medical experts to: * Review execution details for each document. * Annotate and provide feedback to correct hallucinations or omissions. * Download the final validated results in a standardized exchange format for use in the study's evaluation framework.

Administrative Modules

Beyond the session workflow, the platform includes User Management to control expert access and API Configuration to query sessions and results.

3.4 Selecting Smaller Large Language Models (SLM) for the Evaluation

The objective is to identify a set of 5 SLMs that can run locally on consumer-grade hardware while maintaining enough semantic understanding to process (synthetic) clinical texts.

While a comprehensive understanding of general-purpose context may be disregarded, it important that the models demonstrate a robust understanding of clinical context and the ability to perform precise information extraction. Furthermore, our selection criteria for SLMs are not strictly limited to models with specialized medical pre-training. Rather, we aim to investigate the inherent suitability and performance of general-purpose models within this specialized domain.

Because we are evaluating SLMs answers against "Silver/Golden Answers" derived from a larger model (Gemini) and human verification, the selected SLMs must be capable of strict in-

struction following to ensure their outputs can be parsed and scored by our custom evaluation framework. The selection procedure prioritizes models that show “emergent” reasoning capabilities usually reserved for larger models, while remaining compressible enough to fit in local (V)RAM. See: Selection of Prompting Technique: Chain-of-Thought

3.4.1 Procedure: Selection Criteria for ‘suitable’ Clinical SLMs

To filter the hundreds of available open-source models down to a manageable set, we use this five criteria in order.

1. Hardware-Aware Parameter Efficiency

- **Criterion:** Models must have between 7B to 20B paramters that support 4-bit or 8-bit quantization
- **Why:** A standard laptop/desktop with 16Gb Memory (shared or dedicated VRAM) cannot run a 20B model at 16-bit full precision (FP16). For Example:
  - A 7B model requires ~16GB RAM at FP16 but only 5GB to 6GB at 4-bit quantization
  - A 14B model requires ~30GB at FP16 but fits into 10GB to 12GB at 4-bit, making it feasable for profssional consumer desktops
  - Hence a 18B model at 4-bit quantization will still meet the criterion
- **Selection:** Exclude any models <=18B consider choosing higher bit-quantization for smaller models. LLM Model Parameters 2025

2. High Reasoning & Knwledge Benchmark Scores

- **Criterion:** Prioritize models with high scores on MMLU-Pro disciplines Biology, Chemistry, Health and Psychology
- **Why:** Clinical text annotation is not just text generation. It is a reasoning task. Standard benchmarks like MMLU are becoming saturated and less discriminative. MMLU-Pro better distinguish models that “understand” complex topics versus those that just guess.
- **Selection:** Based on the MMLU-Pro Leaderboardsé: Select models that outperform in Biology, Chemistry, Health and Psychology and provide “Reasoning” or “Thinking” to reduce hallucination. See Table below.

3. Instruction Following & Output Structure

- **Criterion:** Select “Instruct” or “Chat” rather than base models

- **Why:** We compare SLM output against Silver/Golden Answers. If the SLM cannot follow instructions, we simply get the output of a “completion engine not an assistant. Base trained models lack of intent recognition.
- **Selection:** Choose the “Instruct” or “Chat” variants

4. Context Window Capacity

- **Criterion:** Minimum context window of 8k tokens (preferably 32k+ or higher)
- **Why:** Clinical notes can be lengthy. If a diagnosis or generally a peatient report exceeds the model’s context window, the model will “forget” early information, leading to missed health infomration annotations. Newer architectures support massive context windows, allowing the model to read a full report in one pass
- **Selection:** Discard models with <8k context limits

5. License & Data Sovereignity

- **Criterion:** Permissive licenses (Apache 2.0, MIT) allowing local commercial use
- **Why:** The primary advantage of SLMs in healthcare is data sovereignty—running locally so patient data never leaves the machine. Open-source models allow to inspect the model and ensure no data is sent to external APIs.
- **Selection:** Model is truly open source (and does not require any API calls)

Proposed set of SLMs for Evaluation

Models	Qualifier	Context Window	License	Size (B)	Remarks
GLM-4-9B-Chat	Chat	128k	MIT	9	
Mistral-Nemo-IT-2407	Instruct	128k	Apache 2.0	12	
Qwen2-7B-Instruct	Instruct	32k	Apache 2.0	7	
Phi-3.5-mini-instruct	Instruct	128k	MIT	3.8	
Llama-3-8B-Instruct	Instruct	8k			
Granite-3.3-2B	Instruct	128k	Apache 2.0	2	Additional to have a really small one

If we want to use models under Llama 3.1, Gemma or Qwen license we need to integrate following NOTICE text:

```
ATTRIBUTION NOTICE

- If using Gemma: "Gemma is provided under and subject to the Gemma Terms of Use found at ai.
- If using Llama 3.1: "Llama 3.1 is licensed under the Llama 3.1 Community License, Copyright

EULA Compliance Template

Section X: AI Usage and Compliance
X.1 License Grant and Pass-Down Terms. Licensor grants Customer a limited license to use the

X.2 Professional Advice Disclaimer. The Software is an automated tool and is NOT a substitute
- Output will not be used as authoritative for the unlicensed practice of medicine, law, or f
- All high-stakes outputs must be reviewed and authorized by a qualified human professional b

X.3 Prohibited Use & Safety. Customer shall not use the Software to:
- Generate or facilitate illegal activities, violence, or terrorism.
- Engage in harassment, bullying, or unlawful discrimination.
- Create malicious code, malware, or viruses.
- Deceive or mislead others, including the creation of disinformation or fake reviews.

X.4 Local Deployment and Liability. As the Software is deployed locally on Customer's private

X.5 Termination for Misuse. [Your Company Name] reserves the right to terminate this Agreemen
```

3.4.2 Evaluation Pipeline

TBD with (kindofwhat?)

3.5 Experimental Setup

3.5.1 Architecture

Maschine von Beni, Chrigels notebook, Google cloud für Gemini, Evaluationsframeworks

3.5.2 Context Engineering Strategies

TBD describe why we used only one prompt

The 3 Most Potent Techniques for Golden Answers (for a state of the art LLM) While Role Prompting and Skeleton-of-Thought are valuable, they are "modifiers" or "accelerators." The three techniques below are architectural necessities for ensuring the accuracy required for a scientific Gold Standard.

1. Prompt Chaining (The Architecture) **Source Evidence:** The Prompt Engineering Guide highlights that chaining is essential for "Document QA" where extraction and synthesis are separate logical steps.

Why it wins

Medical documentation in GraSCCo is unstructured and "messy." Trying to extract, clean, standardize, and format data in a single shot leads to cognitive overload for the model.

Application

You must split the generation of Golden Answers into a pipeline:

- **Step 1 (Extraction):** "Extract all medical entities." (Raw list).
- **Step 2 (Grounding):** "Match these entities to ICD-10/RxNorm codes." (Standardization).
- **Step 3 (Formatting):** "Convert this standardized list into the final JSON schema."

Value

If the JSON is broken, you only debug Step 3. If a drug is missed, you debug Step 1. This traceability is vital for a thesis.

---

2. Self-Consistency (The Validator) **Source Evidence:** The paper *Self-Consistency Improves Chain of Thought Reasoning* proves that replacing greedy decoding with "sample-and-marginalize" (majority voting) improves performance on complex reasoning tasks by significant margins (e.g., +17.9% on GSM8K).

Why it wins

You are creating a "Gold Standard" using AI, which is inherently risky. A single pass of GPT-4 might hallucinate a symptom.

Application

For every GraSCCo document, run your extraction prompt 5 to 10 times.

- **Mechanism:** If 8/10 runs extract “Hypertension,” accept it as Truth. If only 2/10 extract “Diabetes,” discard it as noise/hallucination.

Value

This statistically “purifies” your Golden Answers, making them a defensible ground truth for your scientific evaluation.

---

3. Multi-Persona / Role Prompting (The Expert Layer) **Source Evidence:** New sources from Reddit and K2View emphasize that telling the model who to be (“You are a skeptical expert”) drastically changes the quality of output compared to generic prompts.

Why it wins

GraSCCo contains specific Swiss-German medical shorthand. A generic model might miss context.

Application

Instead of just “Summarize this,” you combine Role Prompting with a Multi-Persona loop:

- **Pass 1 (Persona A - The Scribe):** “You are a medical scribe. Transcribe the key facts.”
- **Pass 2 (Persona B - The Senior Consultant):** “You are a Senior Physician. Review the scribe’s notes against the original text. Point out missing diagnoses or errors.”

Value

This acts as an automated **“Four-Eyes Principle”** (*Vier-Augen-Prinzip*), mimicking the real-world medical workflow where a senior doctor signs off on a junior doctor’s note.

3.6 Evaluation Metrics

3.6.1 Test Setup

llm-validator To facilitate the systematic evaluation described in Phase III and IV, a purpose-built evaluation framework — *llm-validator* — was developed as part of this research. The tool serves as the central instrumentation layer for capturing, executing, and assessing LLM interactions across multiple models and prompting strategies.

Technology Choice. The framework is implemented in Java 21 using the Quarkus application framework, with LangChain4j for LLM integration and an Angular-based web interface for result inspection. The deliberate choice of a JVM-based stack over the more prevalent Python ecosystem in the LLM domain is motivated by the project's alignment with healthcare IT environments: Java remains the dominant technology in enterprise and clinical information systems in Switzerland and the DACH region. By building the evaluation tooling on this stack, the resulting artefact is not only a research instrument but also a reusable component that can be integrated into existing institutional infrastructure without introducing foreign runtime dependencies.

Evaluation Pipeline. The core contribution of the tool lies in its multi-dimensional evaluation pipeline. Test cases — each comprising a clinical query, an optional system prompt, and a golden answer — are organised into *Test Runs* and executed in batch against one or more models. The framework then applies two categories of evaluation metrics:

- **Statistical metrics** (no LLM required): Token-level F1 score, Levenshtein similarity, and embedding-based semantic similarity provide quantitative baselines for output comparison.
- **G-Eval metrics** (LLM-as-a-Judge): Following the G-Eval framework, configurable judge prompts assess *answer relevancy*, *faithfulness*, *hallucination*, and *correctness* against the golden answers established in Phase II. These metrics are stored as database-backed definitions and can be extended without code changes.

Additionally, the system supports *expert evaluation*, allowing a human reviewer to provide qualitative scores — closing the loop between automated assessment and domain expertise.

JSON Structural Similarity A custom metric was developed to assess how well a model's JSON output conforms to the expected schema. The algorithm flattens both the model output and the Silver Answer into leaf-path maps, aligns array elements via greedy best-match, and computes normalised Levenshtein similarity per leaf pair. The overall score is the arithmetic mean across all leaves. For a detailed description see Appendix: JSON Structural Similarity Algorithm.

DAG-Based Medical Extraction Quality To evaluate clinical quality beyond what statistical metrics can capture, a Directed Acyclic Graph (DAG) evaluation metric was developed. Unlike single-prompt LLM-as-a-Judge approaches, the DAG metric decomposes the evaluation into multiple specialised assessment tasks executed by the judge LLM. The medical extraction quality graph evaluates four parallel dimensions — format compliance, factual accuracy, completeness, and medical terminology — and averages their scores. For a detailed description

of the execution engine and the graph structure see Appendix: DAG-Based Medical Extraction Quality Algorithm.

The Logprobs Problem in G-Eval The central mechanism of G-Eval is probability-weighted scoring: rather than taking the LLM's generated score at face value, the token log-probabilities of the score tokens (e.g. "1" through "5") are extracted and a weighted average is computed (Liu u. a. 2023). This approach significantly reduces the known scoring bias of LLMs and is the primary reason for G-Eval's superior human correlation compared to naive LLM-as-a-Judge approaches.

However, the `logprobs` feature that enables this weighted scoring is not uniformly supported across LLM providers. Table **??** summarises the current compatibility landscape.

Tabelle 4: Logprobs compatibility by LLM provider (as of February 2026) {#tab:logprobs-compat}

Provider	Logprobs	Notes
OpenAI (standard models)	Yes	gpt-4o, gpt-4.1-mini etc. via `/v1/chat/completions`
OpenAI (reasoning models)	**No**	o-series, gpt-5-mini — `logprobs` not supported, `temperature` fixed at 1.0
vLLM (self-hosted)	Yes	Any HuggingFace model; logprobs reflect raw model output before post-processing
Together.ai	Yes	Open-weight models via OpenAI-compatible API
Ollama	**No**	Logprobs only on native `/api/generate`, not on OpenAI-compatible `/v1/chat/completions`
LM Studio	**No**	Accepts `top_logprobs` on `/v1/responses` (since v0.3.26, Jan 2026) but returns empty arrays in practice
llama.cpp server	**No**	Returns `null` for logprobs on `/v1/chat/completions`

Particularly problematic is the incompatibility with reasoning models (OpenAI o-series, gpt-5-mini, gpt-5-nano). These models employ an internal reasoning phase that consumes tokens from the `max_completion_tokens` budget before any visible output is produced. For a task that merely requires a single integer score, reasoning models are architecturally unsuitable: they

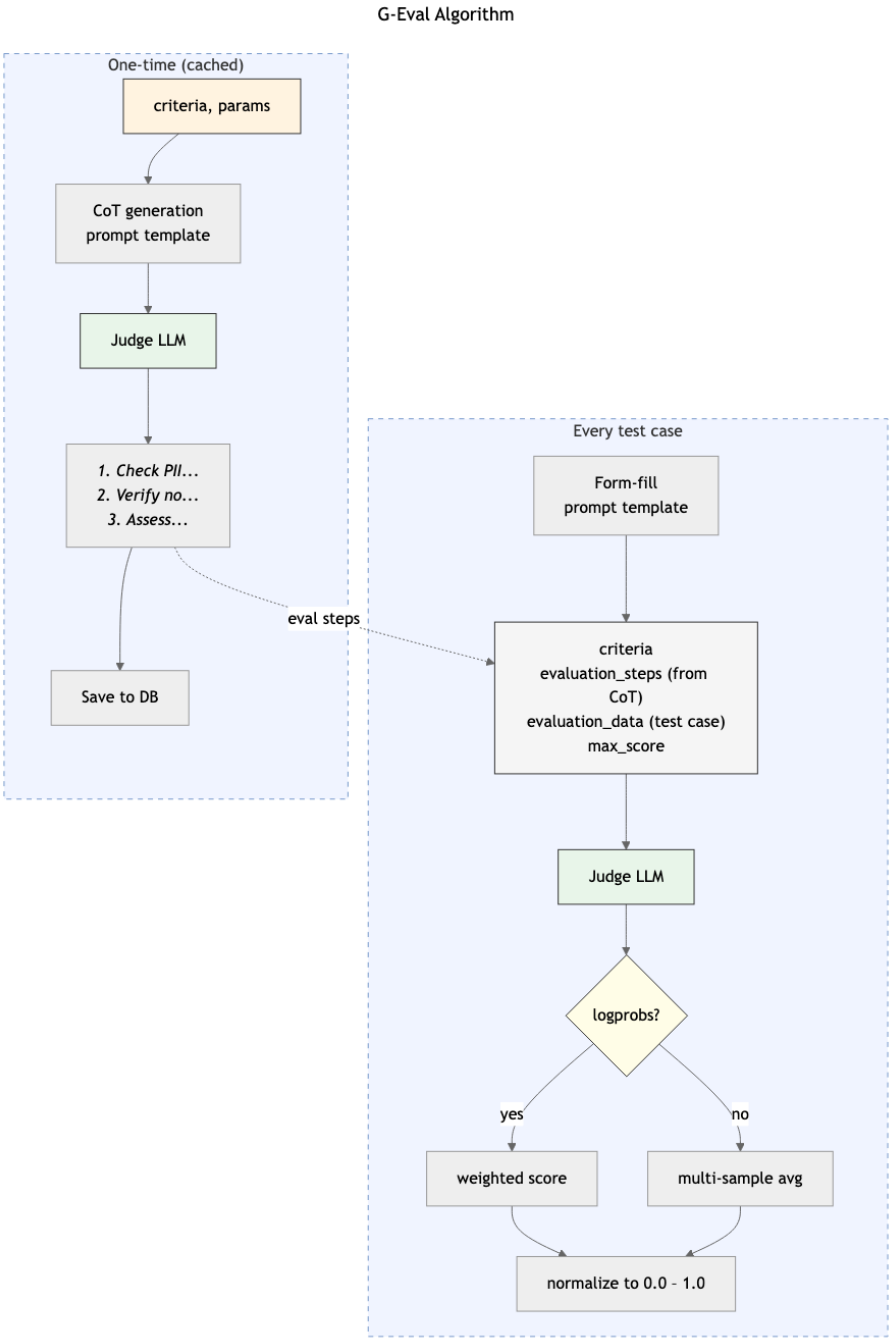


Abbildung 2: G-Eval algorithm: CoT evaluation steps are generated once and cached, then applied per test case with probability-weighted scoring via logprobs or multi-sample fallback.

spend hundreds of tokens on internal deliberation for a trivial decision — while supporting neither `logprobs` nor configurable `temperature` values.

This problem also affects existing reference implementations. DeepEval, the most widely used Python implementation of G-Eval, works around the issue with a hardcoded list of reasoning models for which it falls back to plain JSON extraction without probability weighting — which de facto is no longer G-Eval but a simple LLM-as-a-Judge approach. Several open issues document this limitation: reasoning models break G-Eval entirely[1], custom (non-OpenAI) models never receive weighted summation[2], and the fallback from weighted to unweighted scoring occurs silently without warning[3].

A promising alternative for local execution is vLLM, a high-throughput self-hosted inference engine that provides full logprobs support on its OpenAI-compatible API for any HuggingFace model. While vLLM returns logprobs from the model's raw output (before temperature scaling or penalty adjustments), this is sufficient for G-Eval scoring where the probability distribution over score tokens is the quantity of interest. Due to time constraints we did not engineer vLLM into our evaluation pipeline.

Fallback strategy. For providers without logprobs support, the G-Eval paper defines an alternative method: *multi-sample estimation* with $n = 20$ independent calls at `temperature=1.0`, where each response is parsed for an integer score and the results are averaged (Liu u. a. 2023). This procedure approximates the probability distribution through sampling and thus remains faithful to the G-Eval algorithm — albeit at significantly higher cost (factor 20 compared to the logprobs variant). Practically this turned out to be problematic since this lead to rejected API calls (Vertex AI) or not acceptable performance (local LMStudio).

Consequence for judge model selection. The choice of judge model for G-Eval evaluation is therefore constrained: either a non-reasoning cloud model with logprobs support is used (e.g. gpt-4o-mini), a self-hosted vLLM instance serves as judge, or the cost-intensive multi-sample fallback is required. For this study, an auto-detection strategy is implemented that first attempts the logprobs path and automatically falls back to multi-sample upon failure — enabling both cloud APIs and local models to serve as judges.

4 Results

This chapter presents the empirical findings of the Zero-Shot evaluation run across all nine models and 62 test cases from the GraSCCo corpus. Each test case was evaluated using eight

[1] https://github.com/confident-ai/deepeval/issues/1358
[2] https://github.com/confident-ai/deepeval/issues/1831
[3] https://github.com/confident-ai/deepeval/issues/1029

metrics spanning statistical, embedding-based, and LLM-as-a-Judge categories.

4.1 Overview of Models and Evaluation Metrics

Nine models were evaluated: one large cloud model (Gemini 2.5 Pro via Vertex AI), one small cloud model (GPT-5-nano via OpenAI), and seven locally executable SLMs ranging from 2B to 27B parameters. All models received the identical system prompt and clinical input documents in a Zero-Shot configuration — no few-shot examples or retrieval augmentation was applied.

The evaluation framework applied eight metrics per interaction, categorised into three groups:

- **Statistical metrics** (deterministic, no model required): BLEU, ROUGE, Levenshtein similarity, Token F1, and JSON structural similarity. These measure lexical overlap and structural conformance between the model output and the Silver Answer.
- **Embedding-based metric** (requires an embedding model, but no generative LLM): Semantic similarity, computed via cosine distance on text-embedding-3-small vectors. This metric captures whether the output conveys the same meaning as the Silver Answer, independent of exact wording.
- **LLM-as-a-Judge metrics** (require a generative LLM as evaluator): DAG medical extraction quality and LLM-Judge correctness. These employ an LLM to assess the clinical quality and overall correctness of the extracted content against the Silver Answer.

4.2 Impact of LLM Size

4.2.1 Aggregate Performance

Table **??** presents the mean scores across all 62 test cases per model and metric.

The following three tables present the mean scores grouped by metric category.

Statistical Metrics (deterministic, no model required):

Tabelle 5: Mean statistical metric scores per model across 62 test cases (Zero-Shot). {#tab:avg-scores-stat}

Model	Size	BLEU	ROUGE	Levenshtein	Token F1	JSON Sim.
gemini-2.5-pro	Large	0.122	0.254	0.377	0.380	0.440
gemma3:27b	27B	0.080	0.210	0.343	0.309	0.372
gpt-5-nano	Small	0.085	0.201	0.318	0.313	0.272
granite3.3:2b	2B	0.085	0.165	0.317	0.266	0.254
mistral-nemo	12B	0.080	0.189	0.336	0.275	0.059

Model	Size	BLEU	ROUGE	Levenshtein	Token F1	JSON Sim.
glm4:9b	9B	0.072	0.170	0.322	0.259	0.255
qwen2:7b	7B	0.054	0.159	0.291	0.270	0.160
phi3.5:3.8b	3.8B	0.077	0.115	0.287	0.218	0.103
llama3:8b	8B	0.077	0.160	0.314	0.245	0.000

Embedding-Based Metric (requires embedding model) and **LLM-as-a-Judge Metrics** (require generative LLM as evaluator):

Tabelle 6: Mean embedding and LLM-as-a-Judge scores per model across 62 test cases (Zero-Shot). {#tab:avg-scores-judge}

Model	Size	Sem. Sim.	DAG	LLM-Judge
gemini-2.5-pro	Large	0.835	0.619	0.730
gpt-5-nano	Small	0.861	0.593	0.707
gemma3:27b	27B	0.790	0.520	0.702
granite3.3:2b	2B	0.843	0.528	0.659
qwen2:7b	7B	0.765	0.508	0.675
phi3.5:3.8b	3.8B	0.797	0.479	0.628
glm4:9b	9B	0.732	0.433	0.660
mistral-nemo	12B	0.794	0.261	0.643
llama3:8b	8B	0.650	0.475	0.636

4.2.2 Composite Scores by Metric Category

To provide a consolidated view, Table **??** aggregates the metric averages into three categories and an overall composite score.

Tabelle 7: Composite scores (mean of metric averages) by category. Statistical = 5 lexical/structural metrics; Embedding = semantic similarity (text-embedding-3-small); LLM-as-a-Judge = 2 generative evaluation metrics. {#tab:composite}

Model	Size	Statistical	Embedding	LLM-as-a-Judge	Overall	Avg. Latency (ms)
gemini-2.5-pro	Large (Cloud)	0.315	0.835	0.675	0.470	22’259
gpt-5-nano	Small (Cloud)	0.238	0.861	0.650	0.419	44’443
gemma3:27b	27B	0.263	0.790	0.611	0.416	69’136
granite3.3:2b	2B	0.217	0.843	0.593	0.390	16’506
glm4:9b	9B	0.216	0.732	0.547	0.363	20’237
qwen2:7b	7B	0.187	0.765	0.591	0.360	15’730
phi3.5:3.8b	3.8B	0.160	0.797	0.554	0.338	20’197
mistral-nemo	12B	0.188	0.794	0.452	0.330	24’556

Model	Size	Statistical	Embedding	LLM-as-a-Judge	Overall	Avg. Latency (ms)
llama3:8b	8B	0.159	0.650	0.556	0.320	17'002

Gemini 2.5 Pro achieves the highest overall composite score (0.470), followed by GPT-5-nano (0.419) and Gemma3:27b (0.416). Among the locally executable SLMs, Granite 3.3 (2B) ranks surprisingly high at 0.390 — outperforming several models four to six times its size.

4.2.3 Pass Rates

Table **??** reports the percentage of test cases where each model met or exceeded the metric-specific pass threshold.

Tabelle 8: Pass rates for embedding and LLM-as-a-Judge metrics (%). Statistical lexical metrics (BLEU, ROUGE, Levenshtein, Token F1, JSON structural similarity) are omitted as all models achieve near-0% pass rates on these. Overall includes all 8 metrics. {#tab:pass-rates}

Model	Size	DAG	LLM-Judge	Sem. Sim.	Overall
gemini-2.5-pro	Large	82.3%	90.3%	93.5%	34.3%
gpt-5-nano	Small	83.9%	85.5%	100.0%	33.7%
gemma3:27b	27B	56.5%	77.4%	90.3%	28.4%
granite3.3:2b	2B	59.7%	58.1%	100.0%	27.4%
phi3.5:3.8b	3.8B	50.0%	48.4%	85.5%	23.2%
qwen2:7b	7B	54.8%	64.5%	62.9%	22.8%
mistral-nemo	12B	24.2%	58.1%	74.2%	19.6%
glm4:9b	9B	35.5%	56.5%	37.1%	16.1%
llama3:8b	8B	50.0%	50.0%	8.1%	13.9%

The statistical lexical metrics (BLEU, ROUGE, Levenshtein, Token F1) yield near-zero pass rates across all models — including Gemini 2.5 Pro. This indicates that these thresholds are either too strict for the task or that the extraction task inherently permits semantically equivalent but lexically diverse outputs.

4.3 JSON Structural Compliance

A critical finding concerns the models' ability to produce valid, structurally correct JSON output matching the expected schema. The `json_structural_similarity` metric directly measures this capability.

Tabelle 9: JSON structural similarity distribution per model. {#tab:json-sim}

Model	Mean	Std	Min	Max
gemini-2.5-pro	0.440	0.078	0.272	0.658
gemma3:27b	0.372	0.067	0.193	0.553
glm4:9b	0.255	0.061	0.000	0.386
gpt-5-nano	0.272	0.084	0.000	0.426
granite3.3:2b	0.254	0.076	0.000	0.397
qwen2:7b	0.160	0.101	0.000	0.366
phi3.5:3.8b	0.103	0.136	0.000	0.380
mistral-nemo	0.059	0.117	0.000	0.334
llama3:8b	0.000	0.000	0.000	0.000

Llama3:8b scores 0.000 across all 62 test cases, indicating a complete failure to produce parseable JSON matching the required schema. Mistral-Nemo and Phi3.5 also exhibit high variance with many zero-score cases. In contrast, Gemini 2.5 Pro and Gemma3:27b produce consistently structured output with no zero-score cases.

4.4 Semantic Understanding vs. Format Compliance

A notable divergence emerges between metrics measuring semantic understanding and those measuring structural compliance. Table **??** contrasts the two dimensions.

Tabelle 10: Semantic understanding metrics (left) vs. structural compliance metrics (right). {#tab:semantic-vs-struct}

Model	Semantic Sim.	LLM-Judge	JSON Sim.	BLEU
gpt-5-nano	0.861	0.707	0.272	0.085
granite3.3:2b	0.843	0.659	0.254	0.085
gemini-2.5-pro	0.835	0.730	0.440	0.122
phi3.5:3.8b	0.797	0.628	0.103	0.077
mistral-nemo	0.794	0.643	0.059	0.080
gemma3:27b	0.790	0.702	0.372	0.080
qwen2:7b	0.765	0.675	0.160	0.054
glm4:9b	0.732	0.660	0.255	0.072
llama3:8b	0.650	0.636	0.000	0.077

Most models achieve semantic similarity scores above 0.75, indicating that the extracted medical content is semantically close to the Silver Answers. However, the same models frequently fail to structure this content according to the prescribed JSON schema. This gap is most pronounced for GPT-5-nano (semantic similarity 0.861 vs. JSON similarity 0.272) and Granite 3.3 (0.843 vs. 0.254).

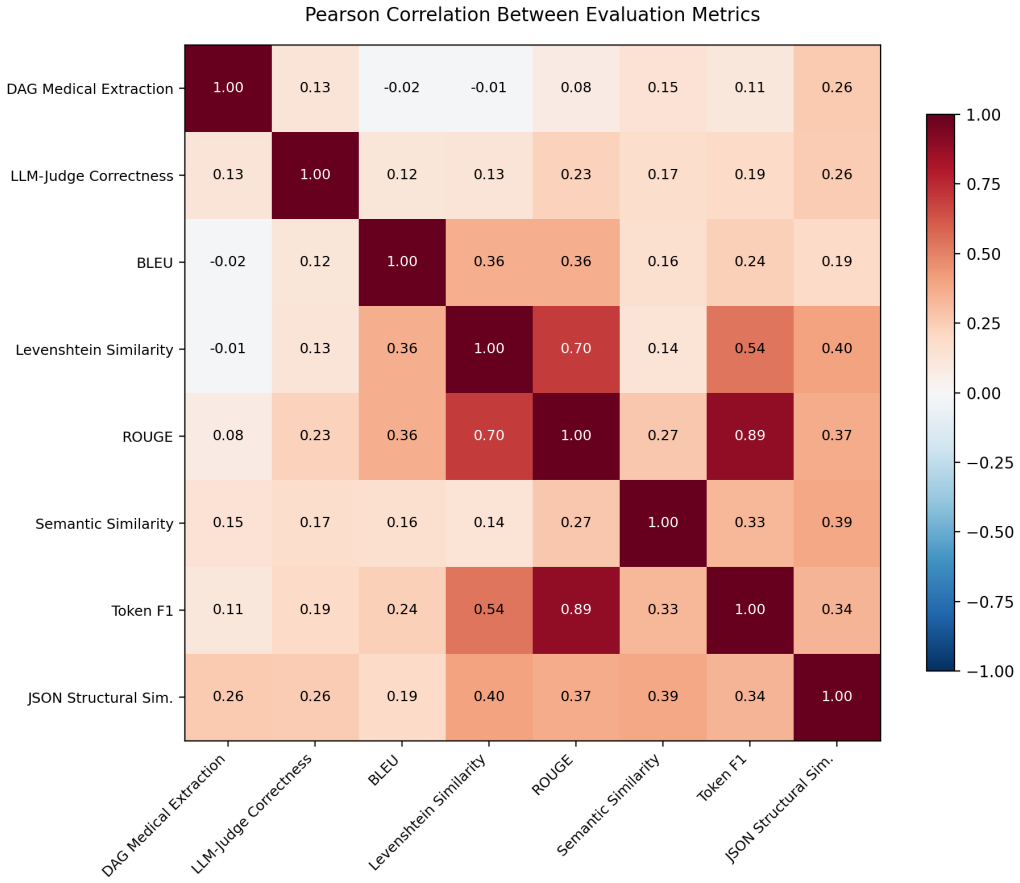


Abbildung 3: Pearson correlation matrix between evaluation metrics. Strong correlations ($r > 0.5$) appear among lexical metrics; LLM-as-a-Judge metrics show low correlation with statistical measures.

4.5 Metric Correlation Analysis

To understand the relationships between evaluation metrics, Figure 3 presents the Pearson correlation matrix computed across all 558 model-document interactions (9 models x 62 test cases).

Several patterns emerge from the correlation analysis:

Strong intra-group correlation among lexical metrics. Levenshtein similarity, ROUGE, and Token F1 form a tightly correlated cluster ($r = 0.54$–0.89). This is expected, as all three measure character- or token-level overlap. BLEU correlates moderately with this group ($r = 0.36$), likely due to its n-gram precision focus versus the recall-oriented nature of ROUGE and Token F1.

Low correlation between LLM-as-a-Judge and statistical metrics. DAG medical extraction quality shows near-zero correlation with the lexical metrics ($r = -0.02$ to 0.15), and LLM-Judge correctness similarly exhibits weak correlations ($r = 0.12$–0.23). This confirms that the LLM-based evaluators capture a fundamentally different quality dimension — clinical extraction fidelity — that lexical overlap metrics cannot approximate.

Semantic similarity occupies a middle ground. While embedding-based semantic similarity correlates weakly with the lexical cluster (r = 0.14–0.33), it also shows only modest correlation with the LLM-as-a-Judge metrics (r = 0.15–0.17). This supports its categorisation as a distinct metric tier: it measures meaning preservation without assessing clinical correctness or structural compliance.

JSON structural similarity is the most independent structural metric. It correlates moderately with the lexical group (r = 0.34–0.40) but also shows the highest correlation with semantic similarity (r = 0.39) among the statistical metrics. This suggests that models producing well-structured JSON also tend to generate more semantically accurate content — format compliance and content quality are not independent.

4.6 Latency

Average inference latency varies considerably across models. The fastest models are Qwen2:7b (15'730 ms) and Granite 3.3:2b (16'506 ms), while Gemma3:27b is the slowest at 69'136 ms per test case. The cloud-based Gemini 2.5 Pro achieves competitive latency (22'259 ms) despite being the largest model, benefiting from optimised cloud infrastructure.

Tabelle 11: Mean inference latency per test case in milliseconds. {#tab:latency}

Model	Mean Latency (ms)	Size
qwen2:7b	15'730	7B
granite3.3:2b	16'506	2B
llama3:8b	17'002	8B
phi3.5:3.8b	20'197	3.8B
glm4:9b	20'237	9B
gemini-2.5-pro	22'259	Large (Cloud)
mistral-nemo	24'556	12B
gpt-5-nano	44'443	Small (Cloud)
gemma3:27b	69'136	27B

5 Discussion

5.1 Interpretation of Results

5.1.1 Model Size Does Not Linearly Predict Performance

The results challenge the assumption that larger models necessarily produce better outputs for clinical extraction tasks. While Gemini 2.5 Pro — the largest model — achieves the highest

composite score (0.470), the ranking among SLMs reveals no consistent correlation between parameter count and output quality. Granite 3.3 with only 2B parameters (composite: 0.390) outperforms GLM4:9b (0.363), Qwen2:7b (0.360), and Phi3.5:3.8b (0.338) — models with two to four times as many parameters. Similarly, Mistral-Nemo at 12B parameters (0.330) ranks second-to-last, performing below the 2B Granite model on both statistical and LLM-as-a-Judge metrics.

This finding suggests that architecture, training data composition, and instruction-tuning quality matter more than raw parameter count for structured medical extraction tasks.

5.1.2 Cloud Models Retain an Advantage — But the Gap Is Narrower Than Expected

GPT-5-nano, despite being marketed as a "nano" model, achieves the highest semantic similarity across all models (0.861) and a 100% pass rate on this metric. It closely matches the 27B Gemma3 model on the overall composite score (0.419 vs. 0.416). This indicates that recent cloud-optimised small models benefit from distillation techniques and training data quality that local open-weight models have not yet matched.

However, the gap between the best cloud model (Gemini: 0.470) and the best local SLM (Gemma3:27b: 0.416) is only 13% — a margin that may be bridgeable through context engineering strategies such as Few-Shot prompting or RAG, which were not applied in this Zero-Shot evaluation.

5.1.3 The Format Compliance Problem

The most critical practical finding is the systematic failure of smaller models to produce structurally valid JSON output. Llama3:8b scores 0.000 on JSON structural similarity across all 62 test cases — it never produces output matching the expected schema. Mistral-Nemo (0.059) and Phi3.5 (0.103) also exhibit severe structural compliance issues.

This has direct implications for clinical deployment: even when a model semantically "understands" the content (as evidenced by high semantic similarity scores), the output cannot be programmatically processed if it does not conform to the expected structure. In a production system, this would require either post-processing heuristics or a format-correction layer — both of which add complexity and potential failure modes.

Notably, Gemma3:27b (0.372) and Gemini 2.5 Pro (0.440) are the only models that consistently produce structured output, suggesting that reliable JSON generation in a Zero-Shot setting may require either larger model capacity or explicit format training.

5.1.4 Semantic Understanding Is Preserved Across Model Sizes

Despite the structural compliance issues, semantic similarity scores remain relatively high across all models (0.650–0.861), with the exception of Llama3:8b. This indicates that even small models (Granite 2B: 0.843, Phi3.5: 0.797) capture the medical content and its meaning to a reasonable degree. The bottleneck is not comprehension but instruction following — specifically, the ability to adhere to a prescribed output format while simultaneously extracting and condensing clinical information.

5.1.5 Mistral-Nemo: An Outlier

Mistral-Nemo (12B) underperforms its parameter class significantly. Its DAG medical extraction quality score (0.261) is the lowest of all models — worse than Granite at 2B (0.528). Combined with its near-zero JSON structural similarity (0.059), this suggests that the model's instruction-following capability for structured extraction in German clinical texts is inadequate despite its size. This reinforces the observation that model selection for domain-specific tasks cannot rely on parameter count alone.

5.2 Implications for Clinical Practice

The results carry several implications for the deployment of local LLMs in clinical settings:

1. **Viability of local deployment:** The semantic understanding scores (>0.75 for most models) demonstrate that local SLMs can extract medically relevant content from clinical documents. This validates the fundamental premise of the DSP4D project — that data-sovereign AI processing on local hardware is feasible.
2. **Format compliance as gating criterion:** For automated pipeline integration (e.g., updating patient records), JSON structural validity is a non-negotiable requirement. Based on the Zero-Shot results, only Gemma3:27b among the local SLMs achieves acceptable structural compliance. Smaller models would require either Few-Shot examples demonstrating the exact output format or a dedicated format-correction step.
3. **Model selection should be task-driven:** The wide variance in performance across models of similar size (e.g., Granite 2B outperforming Mistral-Nemo 12B) indicates that model selection for clinical use cases must be empirically validated rather than inferred from benchmarks or parameter counts.

5.3 Addressing the Research Questions

RQ1: What is the minimum model size for reliable document classification (>95% accuracy)?

No model — including Gemini 2.5 Pro — achieves a 95% pass rate across all metrics. However, when focusing on the clinically most relevant metrics (LLM-Judge correctness and semantic similarity), Gemini 2.5 Pro reaches 90.3% and 93.5% respectively. Among local SLMs, Granite 3.3:2b achieves 100% pass rate on semantic similarity but only 58.1% on LLM-Judge correctness. The 95% threshold is not met by any local model in a Zero-Shot configuration. Context engineering strategies (Few-Shot, RAG) remain to be evaluated.

RQ2: How do different context engineering strategies affect the size-accuracy trade-off?

This evaluation presents the Zero-Shot baseline only. The impact of Few-Shot learning, RAG, and Long-Context strategies on the size-accuracy trade-off is subject to subsequent evaluation phases.

RQ3: Can sub-3B parameter models achieve clinical safety standards with appropriate context?

Granite 3.3 (2B) demonstrates promising semantic comprehension (0.843 similarity, 100% pass rate) but falls short on structural compliance (0.254 JSON similarity). In a Zero-Shot setting, sub-3B models cannot yet meet clinical safety standards across all dimensions. Whether Few-Shot examples can close this gap — particularly for format compliance — is a key question for subsequent phases.

5.4 Limitations

1. **Single prompting strategy:** All results reflect a Zero-Shot configuration. The system prompt used for the SLM evaluation differs from the CoT prompt used for Silver Answer generation (it omits the chain-of-thought reasoning structure). Performance may improve substantially with Few-Shot examples or explicit CoT instructions.
2. **Silver Answer bias:** The reference answers were generated by a large model (Gemini) and only partially validated by a medical expert. As AI-generated Silver Answers — not expert-authored Golden Answers — they carry inherent bias: evaluation metrics favour outputs that resemble the Silver Answer's style and phrasing, which may disadvantage models that express the same medical content differently.
3. **Single evaluation run:** Each model-document combination was executed once. Stochastic variation in model outputs is not captured. Self-Consistency (multiple runs with majority

voting) was not applied.

4. **Hardware heterogeneity:** Local SLMs were executed on a single machine via Ollama, while cloud models used provider-optimised infrastructure. Latency comparisons are therefore not directly comparable across these deployment modes.

5. **German clinical text:** The evaluation is specific to German-language clinical documents from the GraSCCo corpus. Generalisability to other languages or clinical text types is not established.

6. **Metric thresholds:** The near-zero pass rates on lexical metrics (BLEU, ROUGE, Token F1) across all models — including the reference Gemini model — suggest that the pass thresholds for these metrics may be miscalibrated for this extraction task, where semantically equivalent but lexically diverse outputs are expected.

5.5 Future Work

1. **Context engineering evaluation:** Systematic evaluation of Few-Shot, RAG, and Long-Context strategies across all models to quantify the impact of context engineering on the size-accuracy trade-off — particularly for format compliance in small models.

2. **Prompt optimisation for SLMs:** The current system prompt was designed for a SOTA model. Adapting prompts specifically for smaller models (e.g., simpler instructions, explicit format examples) may yield disproportionate improvements.

3. **Self-Consistency runs:** Applying the multi-sample estimation approach (5–10 runs per document) to assess output stability and reduce stochastic variance.

4. **Expanded expert validation:** Involving multiple medical experts to elevate Silver Answers to true Golden Answers and reduce single-annotator bias.

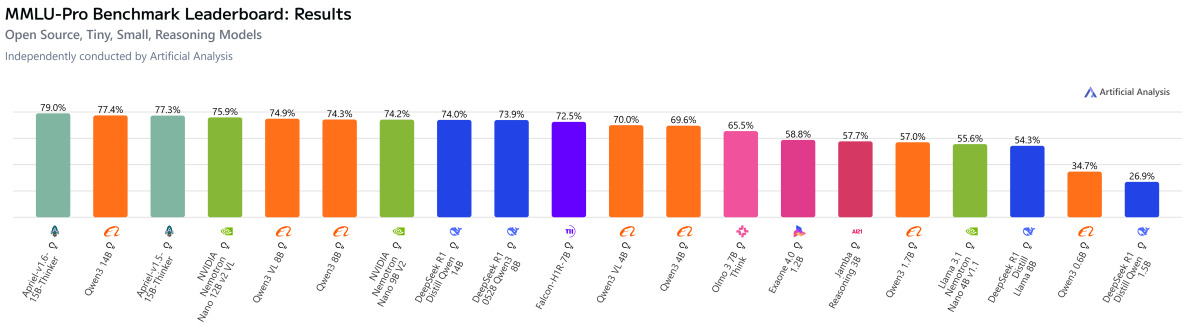
5. **Fine-tuning exploration:** Investigating whether task-specific fine-tuning of small models (2–7B) on a subset of validated reference answers can close the format compliance gap observed in Zero-Shot evaluation.

6 Appendices

6.1 MMLU-Pro Benchmark Leaderboard

MMLU-Pro Benchmark Leaderboard (filtered)

Filter: Size Class: Tiny, Small; Open Weights: Open Source; Reasoning: Reasoning; Paramters Count: <=18B



This “Gold Standard” example now includes the **Internal Monologue**, which is the hallmark of the Chain of Thought (CoT) approach. It demonstrates how the model “thinks” through the German syntax before committing to the structured fields.

6.2 Appendix: Gold Standard Example (CoT Approach)

Below is the processing of the same kardiologische report, but utilizing the **Klinische Analyse** to ensure clinical accuracy.

6.2.1 Input: Sample Clinical Report (GraSCCo-Style)

Bericht vom 14.03.2025 – Kardiologische Gemeinschaftspraxis Dr. Müller

Anamnese: 68-jähriger Patient stellt sich zur Verlaufskontrolle bei bekannter arterieller Hypertonie und stabiler Angina Pectoris vor. Aktuell klagt der Patient über gelegentliches Herzstolpern.

Befunde: RR 145/90 mmHg, Puls 72/min (arrhythmisch). EKG zeigt neu aufgetretenes Vorhofflimmern. Labor: LDL 110 mg/dl, Kreatinin 0,9 mg/dl.

Medikation: Bisher Bisoprolol 5mg 1-0-0 und ASS 100mg 0-1-0.

Beurteilung: Umstellung der Antikoagulation aufgrund des Vorhofflimmerns notwendig.

Plan: Absetzen von ASS 100mg. Neuansetzung von Apixaban 5mg 1-0-1. Echokardiographie in 2 Wochen zur Klärung der Klappenfunktion.

6.2.2 Output: Golden Answer (CoT)

6.2.3 Klinische Analyse (Internal Monologue)

1. **Datum/Quelle:** Der Bericht ist datiert auf den 14.03.2025. Aussteller ist Dr. Müller (Kardiologie).
2. **Kategorisierung:** Da es sich um eine kardiologische Praxis handelt und Herzrhythmusstörungen (Vorhofflimmern) sowie Hypertonie behandelt werden, sind “Kardiologie” und “Innere Medizin” die passenden Fachbereiche.
3. **Diagnosen:** Bekannt sind Hypertonie und Angina Pectoris. Neu hinzu kommt “Vorhofflimmern” (dokumentiert via EKG).
4. **Messwerte:** RR (145/90), Puls (72, arrhythmisch), LDL (110) und Kreatinin (0,9) sind explizit gelistet.
5. **Medikation-Logik:** * *Status Quo:* Bisoprolol und ASS 100mg.

   - *Änderung:* Der Text sagt “Absetzen von ASS” und “Neuansetzung von Apixaban”. Bisoprolol wird nicht erwähnt, bleibt also Teil der aktuellen Medikation.
   - *Wichtig:* Apixaban ist die neue Empfehlung aufgrund des Vorhofflimmerns.

6. **Follow-up:** Der Plan sieht eine Echokardiographie in 2 Wochen vor.

6.2.4 Structured Health Record Update

1. **Categories:** Kardiologie, Innere Medizin
2. **Date and Source:** 14.03.2025; Kardiologische Gemeinschaftspraxis Dr. Müller
3. **Diagnosis:** Arterielle Hypertonie, stabile Angina Pectoris, neu aufgetretenes Vorhofflimmern (EKG-gesichert)
4. **Relevant metrics:** RR: 145/90 mmHg; Puls: 72/min (arrhythmisch); LDL: 110 mg/dl; Kreatinin: 0,9 mg/dl
5. **Current or adviced medications:** * **Aktuelle Medikation:** Bisoprolol 5mg (1-0-0)

   - **Empfohlene/Verschriebene Medikation:** Apixaban 5mg (1-0-1); *Hinweis: ASS 100mg wurde abgesetzt.*

6. **Follow up:** Echokardiographie in 2 Wochen zur Klärung der Klappenfunktion.

6.2.5 Evaluation of the CoT Benefit

As seen in section **5**, the CoT process allowed the model to explicitly realize that **ASS 100mg** is no longer “Current” but is part of the “Change” logic. Without CoT, a model might simply list all three drugs under “Current” because they all appear in the text.

6.3 Appendix: Drilldown for SLM Selection for Evaluation

6.3.1 Selection Steps 1. - 3.

MMLU-Pro Leaderboard (filtered)

Evaluation Based on benchmarks in disciplines: Biology, Chemistry, Health and Psychology and maximum parameters 18B, Qualifier: Chat or Instruct Models:

Models	Qualifier	Model Size (B)	Data Source	Overall	Biology	Chemistry	Health	Psychology	Average selected Disciplines
Gemma-2-9B-it	Instruct	9	TIGER-Lab	0.5208	0.7587	0.4664	0.5844	0.6617	0.6178
GLM-4-9B-Chat	Chat	9	TIGER-Lab	0.4801	0.7015	0.4117	0.5379	0.6165	0.5669
EXAONE-3.5-7.8B-Instruct	Instruct	7.8	TIGER-Lab	0.4624	0.7308	0.3719	0.4707	0.5965	0.542475
Mistral-Nemo-Instruct-2407	Instruct	12	TIGER-Lab	0.4481	0.6583	0.3445	0.5281	0.6165	0.53685
Qwen2-7B-Instruct	Instruct	7	TIGER-Lab	0.4724	0.6625	0.3772	0.4645	0.6128	0.52925
Llama-3.1-8B-Instruct	Instruct	8	TIGER-Lab	0.4425	0.6304	0.3763	0.5073	0.6003	0.528575
Yi-1.5-9B-Chat	Chat	9	TIGER-Lab	0.4595	0.6667	0.3949	0.4352	0.594	0.5227

Models	Qualifier	Model Size (B)	Data Source	Overall	Biology	Chemistry	Health	Psychology	Average selected Disciplines
Phi-3.5-mini-instruct	Instruct	3.8	TIGER-Lab	0.4787	0.7057	0.4125	0.5244	0.4188	0.51535
Llama-3-8B-Instruct	Instruct	8	TIGER-Lab	0.4098	0.6653	0.28	0.4902	0.594	0.507375
Granite-3.1-8B-Instruct	Instruct	8	TIGER-Lab	0.4103	0.5746	0.3145	0.4707	0.5739	0.483425
EXAONE-3.5-2.4B-Instruct	Instruct	2.4	TIGER-Lab	0.391	0.6541	0.3171	0.3851	0.5038	0.465025
Qwen1.5-14B-Chat	Chat	14	TIGER-Lab	0.3802	0.6151	0.2615	0.4218	0.5251	0.455875
Ministral-8B-Instruct	Instruct	8	TIGER-Lab	0.3793	0.59	0.2641	0.4328	0.5163	0.4508
Yi-1.5-6B-Chat	Chat	6	TIGER-Lab	0.3823	0.5746	0.3074	0.3362	0.5013	0.429875
DeepSeek-Coder-V2-Lite-IT	Instruct	16	TIGER-Lab	0.4157	0.5007	0.4293	0.2995	0.4687	0.42455
Deepseek-Math-7B-Instruct	Instruct	7	TIGER-Lab	0.353	0.46	0.4108	0.2506	0.3947	0.379025
Granite-3.1-2B-Instruct	Instruct	2	TIGER-Lab	0.3197	0.5007	0.2412	0.3056	0.4411	0.37215

6.3.2 Selection Step 4.

Context windows >=8k

Models	Qualifier	Context Window	License	Size (B)	Overall	Biology	Chemistry	Health	Psychology	Average selected Disciplines
Gemma-2-9B-it	Instruct	8k	Gemma Terms	9	0.5208	0.7587	0.4664	0.5844	0.6617	0.6178
GLM-4-9B-Chat	Chat	128k	MIT	9	0.4801	0.7015	0.4117	0.5379	0.6165	0.5669
EXAONE-3.5-7.8B-IT	Instruct	32k	EXAONE NC	7.8	0.4624	0.7308	0.3719	0.4707	0.5965	0.542475
Mistral-Nemo-IT-2407	Instruct	128k	Apache 2.0	12	0.4481	0.6583	0.3445	0.5281	0.6165	0.53685
Qwen2-7B-Instruct	Instruct	32k	Apache 2.0	7	0.4724	0.6625	0.3772	0.4645	0.6128	0.52925
Llama-3.1-8B-IT	Instruct	128k	Llama 3.1	8	0.4425	0.6304	0.3763	0.5073	0.6003	0.528575
Phi-3.5-mini-instruct	Instruct	128k	MIT	3.8	0.4787	0.7057	0.4125	0.5244	0.4188	0.51535
Llama-3-8B-Instruct	Instruct	8k	Llama 3	8	0.4098	0.6653	0.28	0.4902	0.594	0.507375
Granite-3.1-8B-IT	Instruct	128k	Apache 2.0	8	0.4103	0.5746	0.3145	0.4707	0.5739	0.483425
EXAONE-3.5-2.4B-IT	Instruct	32k	EXAONE NC	2.4	0.391	0.6541	0.3171	0.3851	0.5038	0.465025
Qwen1.5-14B-Chat	Chat	32k	Apache 2.0	14	0.3802	0.6151	0.2615	0.4218	0.5251	0.455875
Ministral-8B-Instruct	Instruct	128k	Mistral	8	0.3793	0.59	0.2641	0.4328	0.5163	0.4508

Models	Qualifier	Context Window	License	Size (B)	Overall	Biology	Chemistry	Health	Psychology	Average selected Disciplines
DeepSeek-Coder-V2-IT	Instruct	128k	DeepSeek	16	0.4157	0.5007	0.4293	0.2995	0.4687	0.42455
Granite-3.1-2B-IT	Instruct	128k	Apache 2.0	2	0.3197	0.5007	0.2412	0.3056	0.4411	0.37215

6.3.3 Selection Step 5.

Remaining Permissive Open-Source models:

Models	Qualifier	Context Window	License	Size (B)	Overall	Biology	Chemistry	Health	Psychology	Average selected Disciplines
GLM-4-9B-Chat	Chat	128k	MIT	9	0.4801	0.7015	0.4117	0.5379	0.6165	0.5669
Mistral-Nemo-IT-2407	Instruct	128k	Apache 2.0	12	0.4481	0.6583	0.3445	0.5281	0.6165	0.53685
Qwen2-7B-Instruct	Instruct	32k	Apache 2.0	7	0.4724	0.6625	0.3772	0.4645	0.6128	0.52925
Phi-3.5-mini-instruct	Instruct	128k	MIT	3.8	0.4787	0.7057	0.4125	0.5244	0.4188	0.51535
Llama-3-8B-Instruct	Instruct	8k	Llama 3	8	0.4098	0.6653	0.28	0.4902	0.594	0.507375
Granite-3.1-8B-IT	Instruct	128k	Apache 2.0	8	0.4103	0.5746	0.3145	0.4707	0.5739	0.483425
Qwen1.5-14B-Chat	Chat	32k	Apache 2.0	14	0.3802	0.6151	0.2615	0.4218	0.5251	0.455875

Models	Qualifier	Context Window	License	Size (B)	Overall	Biology	Chemistry	Health	Psychology	Average selected Disciplines
DeepSeek-Coder-V2-IT	Instruct	128k	DeepSeek	16	0.4157	0.5007	0.4293	0.2995	0.4687	0.42455
Granite-3.1-2B-IT	Instruct	128k	Apache 2.0	2	0.3197	0.5007	0.2412	0.3056	0.4411	0.37215

Comparison of Licenses used remainging from Step 4:

Model Family	Specific Licenses Mentioned	Open Source?	Commercial Use	External API Required?	Key Restrictions & Notes
Permissive	MIT / Apache 2.0	Yes	Unrestricted	No (Local execution)	“Do whatever you want” licenses; zero downstream legal obligations.
Gemma	Gemma Terms of Use	No (Open Weights)	Permitted	No (Local weights available)	Prohibits use for unlicensed professional advice or violating safety policies.
Llama (3.x)	Llama 3.1 Community License	No (Open Weights)	Permitted (with caps)	No (Local execution supported)	Requires a special license if users exceed 700M monthly active users.
DeepSeek	DeepSeek Model License	No (Open Weights)	Permitted	No (Local weights available)	Allows modifications and derivative works, including model distillation.
EXAONE	EXAONE Non-Commercial (NC)	No	Prohibited	No	Strictly restricted to research and experimental purposes only.
Mistral	Mistral AI Non-Production / Commercial	No	Restricted / Tiered	Optional (API vs local)	Smaller models are often Apache 2.0; flagship models require commercial agreements.

6.4 Appendix: JSON Structural Similarity Algorithm

A custom metric was developed to assess how well a model's JSON output conforms to the expected schema, independent of content correctness. Unlike lexical metrics (BLEU, ROUGE) that treat the output as flat text, this metric operates on the JSON structure itself.

The algorithm proceeds in four steps:

1. **JSON extraction and flattening.** Both the model output and the Silver Answer are pre-processed: markdown code fences are stripped, and the JSON content is extracted by bracket-matching. The resulting JSON objects are then recursively flattened into leaf-path maps using dot notation for objects and bracket notation for arrays (e.g., `structured_health_record.medi` or `structured_health_record.categories[0]`). Configurable paths (e.g., `internal_monologue`) can be excluded from comparison.

2. **Array alignment.** JSON arrays pose a challenge because the model may output the same elements in a different order. The algorithm identifies array base paths, groups their children by index, and performs greedy best-match alignment: for each element in the Silver Answer's array, the response element with the highest average Levenshtein similarity across shared sub-fields is selected as the match.

3. **Leaf-by-leaf comparison.** For each aligned path, the leaf values are compared using normalised Levenshtein similarity ($1 - \frac{d(s_1,s_2)}{\max(|s_1|,|s_2|)}$). Missing fields in the response score 0.0; matching `null` values on both sides score 1.0.

4. **Aggregation.** The overall score is the arithmetic mean across all leaf-pair similarities. Sub-scores are also computed per top-level key, enabling inspection of which sections (e.g., `diagnosis` vs. `medications`) the model handles well or poorly.

A score of 0.0 indicates either that the model output could not be parsed as JSON at all, or that none of the expected paths were present. A score of 1.0 indicates perfect structural and content match at every leaf.

6.4.1 Core Implementation

JSON flattening (recursive leaf-path extraction):

```
private static void flattenNode(String prefix, JsonNode node,
        Map<String, String> result, int depth) {
    if (node.isObject()) {
```

```
        Iterator<Map.Entry<String, JsonNode>> fields = node.fields();
        while (fields.hasNext()) {
            var entry = fields.next();
            String childPrefix = prefix.isEmpty()
                ? entry.getKey() : prefix + "." + entry.getKey();
            flattenNode(childPrefix, entry.getValue(), result, depth + 1);
        }
    } else if (node.isArray()) {
        for (int i = 0; i < node.size(); i++) {
            flattenNode(prefix + "[" + i + "]", node.get(i), result, depth + 1);
        }
    } else if (node.isNull()) {
        result.put(prefix, null);
    } else {
        result.put(prefix, node.asText());
    }
}
```

Leaf comparison and aggregation (per-path Levenshtein similarity):

```
for (var entry : alignedGolden.entrySet()) {
    String path = entry.getKey();
    String goldenValue = entry.getValue();
    String responseValue = alignedResponse.get(path);

    if (responseValue == null && goldenValue == null) {
        leafScores.put(path, 1.0);
    } else if (responseValue == null) {
        leafScores.put(path, 0.0);
        missingInResponse.add(path);
    } else {
        leafScores.put(path,
            levenshteinSimilarity(goldenValue, responseValue));
    }
}

double overallScore = leafScores.values().stream()
        .mapToDouble(Double::doubleValue)
        .average().orElse(0.0);
```

6.5 Appendix: DAG-Based Medical Extraction Quality Algorithm

To evaluate the clinical quality of model outputs beyond what statistical metrics can capture, a Directed Acyclic Graph (DAG) evaluation metric was developed. Unlike single-prompt LLM-as-a-Judge approaches that ask one broad question, the DAG metric decomposes the evaluation into multiple specialised assessment tasks — each executed by the judge LLM — and aggregates their results through a graph of conditional judgements.

6.5.1 DAG Execution Engine

The DAG execution engine traverses a graph of four node types:

- **Task nodes** present the judge LLM with specific evaluation instructions and a subset of the evaluation context (actual output, expected output, original input). The LLM's response is stored as accumulated context for downstream nodes.
- **Binary judgement nodes** ask the judge LLM a yes/no question based on accumulated context. The result determines which branch (true/false child) is followed — enabling conditional evaluation paths.
- **Non-binary judgement nodes** present the judge LLM with multiple verdict options (e.g., "fully compliant", "minor issues", "significant issues"). The LLM selects the most appropriate verdict, routing to the corresponding child node.
- **Verdict nodes** are terminal leaves that carry a predefined numeric score (0.0–1.0).

When a graph has multiple root nodes, their branches are executed in parallel (using virtual threads) and the final score is the arithmetic mean of all branch scores. A short-circuit mechanism terminates evaluation early if any branch returns 0.0.

6.5.2 Medical Extraction Quality Graph

The specific DAG graph used for the `dag_medical_extraction_quality` metric evaluates four parallel dimensions:

1. **Format compliance** (structural branch): A task node analyses the output for JSON validity, presence of required top-level keys (`internal_monologue`, `structured_health_record`), and schema conformance. A binary judgement then splits:

- If valid JSON: a non-binary judgement rates schema compliance as *fully compliant* (1.0), *minor issues* (0.7), or *significant issues* (0.3).
- If invalid JSON: a non-binary judgement classifies the output as *recoverable* (0.15) or *garbage* (0.0).

2. **Factual accuracy** (content branch): A task node performs field-by-field comparison against the Silver Answer, marking each field as CORRECT, PARTIALLY_CORRECT, MISSING, or HALLUCINATED. A non-binary judgement then rates overall accuracy as *highly accurate* (1.0), *mostly accurate* (0.75), *partially accurate* (0.4), or *inaccurate* (0.1). Hallucinations are penalised most severely.
3. **Completeness** (coverage branch): A task node counts populated vs. expected fields. A non-binary judgement rates completeness as *complete* (1.0), *mostly complete* (0.7), or *incomplete* (0.3).
4. **Medical terminology** (language branch): A task node evaluates whether the output uses professional medical shorthand, standard abbreviations, and maintains language consistency with the input document. A non-binary judgement rates terminology as *excellent* (1.0), *adequate* (0.6), or *poor* (0.2).

The final score is the average of all four branch scores, yielding a value between 0.0 and 1.0 that captures format compliance, factual fidelity, extraction completeness, and domain-appropriate language in a single metric.

6.5.3 Core Implementation

Node dispatch (sealed interface with pattern matching):

```
private DAGExecutionResult executeNode(DAGNode node,
        EvaluationContext context,
        Map<String, String> accumulatedContext,
        List<TraceEntry> trace) {
    return switch (node) {
        case TaskNode task ->
            executeTask(task, context, accumulatedContext, trace);
        case BinaryJudgementNode binary ->
            executeBinaryJudgement(binary, context,
                accumulatedContext, trace);
        case NonBinaryJudgementNode nonBinary ->
            executeNonBinaryJudgement(nonBinary, context,
```

```
                accumulatedContext, trace);
        case VerdictNode verdict ->
            executeVerdict(verdict, context,
                accumulatedContext, trace);
    };
}
```

Parallel branch execution (virtual threads with short-circuit on zero score):

```
private DAGExecutionResult executeParallelBranches(
        List<DAGNode> branches, EvaluationContext context,
        Map<String, String> accumulatedContext,
        List<TraceEntry> trace) {
    var executor = Executors.newVirtualThreadPerTaskExecutor();
    List<CompletableFuture<DAGExecutionResult>> futures =
        branches.stream()
            .map(branch -> CompletableFuture.supplyAsync(() ->
                executeNode(branch, context,
                    new HashMap<>(accumulatedContext),
                    new ArrayList<>()), executor))
            .toList();

    CompletableFuture.allOf(
        futures.toArray(new CompletableFuture[0])).join();

    double totalScore = 0.0;
    int count = 0;
    for (var future : futures) {
        DAGExecutionResult result = future.join();
        trace.addAll(result.trace());
        if (result.score() == 0.0) {
            return new DAGExecutionResult(0.0, trace, true);
        }
        totalScore += result.score();
        count++;
    }
    return new DAGExecutionResult(
        count > 0 ? totalScore / count : 0.0, trace, false);
}
```

List of Figures

List of Tables

Glossary

Context Engineering The practice of designing prompts and providing relevant information to improve LLM performance on specific tasks.

Edge Deployment Running machine learning models locally on devices rather than in the cloud.

Few-Shot Learning Providing a small number of examples in the prompt to guide model behavior.

GraSCCo Graz Synthetic Clinical text Corpus — a German clinical text corpus for NLP research.

RAG (Retrieval-Augmented Generation) A technique that combines information retrieval with text generation to improve accuracy.

References

Alsentzer, Emily, John Murphy, Willie Boag, Wei-Hung Weng, Di Jindi, Alistair Johnson, und Matthew McDermott. 2019. «Publicly Available Clinical BERT Embeddings». *arXiv preprint arXiv:1904.03323*.

Banerjee, Satanjeev, und Alon Lavie. 2005. «METEOR: An automatic metric for MT evaluation with improved correlation with human judgments». In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 65–72.

Carlini, Nicholas, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, u. a. 2021. «Extracting Training Data from Large Language Models». In *30th USENIX Security Symposium*, 2633–50.

Es, Shahul, Jithin James, Luis Espinosa Anke, und Steven Schockaert. 2024. «Ragas: Automated Evaluation of Retrieval Augmented Generation». In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*.

Fu, Jinlan, See-Kiong Ng, Zhengbao Jiang, und Pengfei Liu. 2024. «GPTScore: Evaluate as You Desire». In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Guluzade, Aynur, Naguib Heiba, Zeyd Boukhers, Florim Hamiti, Jahid Hasan Polash, Yehya

Mohamad, und Carlos Velasco Nunez. 2025. «ELMTEX: Fine-Tuning Large Language Models for Structured Clinical Information Extraction. A Case Study on Clinical Reports». *arXiv preprint arXiv:2502.05638*.
Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, u. a. 2022. «Training Compute-Optimal Large Language Models». *arXiv preprint arXiv:2203.15556*.
Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, und Dario Amodei. 2020. «Scaling Laws for Neural Language Models». *arXiv preprint arXiv:2001.08361*.
Kim, Seungone, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, u. a. 2024. «Prometheus 2: An Open Source Language Model Specialized in Evaluating Other Language Models». *arXiv preprint arXiv:2405.01535*.
Kusner, Matt, Yu Sun, Nicholas Kolkin, und Kilian Weinberger. 2015. «From word embeddings to document distances». In *International conference on machine learning*, 957–66.
Levenshtein, Vladimir I. 1966. «Binary codes capable of correcting deletions, insertions, and reversals». *Soviet physics doklady* 10 (8): 707–10. https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf.
Lin, Chin-Yew. 2004. «Rouge: A package for automatic evaluation of summaries». In *Text summarization branches out*, 74–81.
Liu, Yang, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, und Chenguang Zhu. 2023. «G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment». In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.
Lohr, Christina, Franz Matthies, Jakob Faller, Luise Modersohn, Andrea Riedel, Udo Hahn, Martin Boeker, u. a. 2025. «GraSCCo_PII_V2 - Graz Synthetic Clinical text Corpus with PII Annotations». https://doi.org/10.5281/zenodo.15747389.
Lu, Zhenyan, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Wei Liu, Jian Luan, Xiwen Zhang, Nicholas D Lane, und Mengwei Xu. 2025. «Demystifying Small Language Models for Edge Deployment». https://aclanthology.org/2025.acl-long.718.pdf.
Lu, Zhenyan, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D Lane, und Mengwei Xu. 2024. «Small Language Models: Survey, Measurements, and Insights». *arXiv preprint arXiv:2409.15790*.
Manning, Christopher D, Prabhakar Raghavan, und Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf.
Modersohn, Luise, Stefan Schulz, Christina Lohr, und Udo Hahn. 2022. «GRASCCO—The First Publicly Shareable, Multiply-Alienated German Clinical Text Corpus». In *German Medical Data Sciences 2022–Future Medicine: More Precise, More Integrative, More Sustainable!*, 66–72. IOS Press. https://doi.org/10.3233/SHTI220805.
Panickssery, Arjun, Samuel R Bowman, und Shi Feng. 2024. «LLM Evaluators Recognize and Favor Their Own Generations». *arXiv preprint arXiv:2404.13076*.

Papineni, Kishore, Salim Roukos, Todd Ward, und Wei-Jing Zhu. 2002. «Bleu: a method for automatic evaluation of machine translation». In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–18.

Reiter, Ehud. 2018. «A structured review of the validity of BLEU». *Computational Linguistics* 44 (3): 393–401. https://aclanthology.org/J18-3002.pdf.

Sainz, Oscar, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, und Eneko Agirre. 2024. «Data Contamination Quiz: A Tool to Detect and Estimate Contamination in Large Language Models». *arXiv preprint arXiv:2311.06233*.

Saito, Keita, Akifumi Wachi, Koki Wataoka, und Youhei Akimoto. 2024. «Verbosity Bias in Preference Labeling by Large Language Models». *arXiv preprint arXiv:2310.10076*.

Singh, Tanmay, Harshvardhan Aditya, Vijay K. Madisetti, und Arshdeep Bahga. 2024. «Whispered Tuning: Data Privacy Preservation in Fine-Tuning LLMs through Differential Privacy». *Journal of Software Engineering and Applications*. https://doi.org/10.4236/jsea.2024.171001.

Sokolova, Marina, und Guy Lapalme. 2009. «A systematic analysis of performance measures for classification tasks». *Information Processing & Management* 45 (4): 427–37. https://doi.org/10.1016/j.ipm.2009.03.002.

Wu, Jiajun, Swaleh Zaidi, Braden Teitge, Henry Leung, Jiayu Zhou, Jessalyn Holodinsky, und Steve Drew. 2025. «Dual-stage and Lightweight Patient Chart Summarization for Emergency Physicians». *arXiv preprint arXiv:2510.06263*.

Xiao, Chaojun, Jie Cai, Weilin Zhao, Guoyang Zeng, Biyuan Lin, Jie Zhou, Zhi Zheng, Xu Han, Zhiyuan Liu, und Maosong Sun. 2024. «Densing Law of LLMs». *arXiv preprint arXiv:2412.04315*.

Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q Weinberger, und Yoav Artzi. 2020. «BERTScore: Evaluating Text Generation with BERT». In *International Conference on Learning Representations*.

Zhao, Wei, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, und Steffen Eger. 2019. «MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance». In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 563–78. https://aclanthology.org/D19-1053.pdf.

Zhong, Xian, S Li, Z Chen, L Ge, D Yu, S Wang, L You, und H Shang. 2025. «Considerations for Patient Privacy of Large Language Models in Health Care: Review». *Journal of Medical Internet Research*. https://doi.org/10.2196/76571.

Zou, Andy, Zifan Wang, J Zico Kolter, und Matt Mattjung. 2023. «Universal and Transferable Adversarial Attacks on Aligned Language Models». *arXiv preprint arXiv:2307.15043*.

Selbständigkeitserklärung

Ich bestätige, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Textstellen, die nicht von mir stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ich bestätige weiterhin, dass ich bei der Erstellung dieser Studienarbeit durchgehend steuernd gearbeitet habe und von einer KI erzeugte Texte bzw. Textfragmente nicht unreflektiert übernommen habe.

Ort, Datum: Unterschrift: