DocUA at CRF Filling 2026: LLM StructCore — Schema-Guided Reasoning Condensation and Deterministic Compilation

Serhii Zabolotnii
Cherkasy State Business College
Cherkasy, Ukraine
zabolotnii.serhii@csbc.edu.ua

Abstract

Automatically filling Case Report Forms (CRFs) from clinical notes is challenging due to noisy language, strict output contracts, and the high cost of false positives. We describe our CL4Health 2026 submission for Dyspnea CRF filling (134 items) using a *contract-driven* two-stage design grounded in *Schema-Guided Reasoning* (SGR) (Abdullin, 2026). The key task property is **extreme sparsity**: the majority of fields are `unknown`, and official scoring penalizes both empty values and unsupported predictions. We shift from a single-step "LLM predicts 134 fields" approach to a decomposition where (i) Stage 1 produces a stable SGR-style JSON summary with exactly 9 domain keys, and (ii) Stage 2 is a fully deterministic, 0-LLM compiler that parses the Stage 1 summary, canonicalizes item names (optionally using a UMLS alias map with 134/134 coverage), normalizes predictions to the official controlled vocabulary (13 categories), applies evidence-gated false-positive filters, and expands the output into the required 134-item format. On the dev80 split, the best teacher configuration (Mistral Large 3 Stage 1 $\rightarrow$ Stage 2 deterministic) achieves macro-F1 **0.6543** (EN) and **0.6905** (IT); on the hidden test200, the submitted English variant scores **0.63** on Codabench. The pipeline is language-agnostic: Italian results match or exceed English with no language-specific engineering.

Keywords: clinical NLP, information extraction, schema-guided reasoning, controlled vocabulary, reproducibility, edge AI

1. Introduction

The CL4Health 2026 CRF filling shared task requires mapping unstructured clinical notes to a strict 134-item Dyspnea CRF. In practice, end-to-end LLM extraction often fails either due to output format drift (invalid JSON / JSONL structure) or due to unsupported "hallucinated" fills that increase false positives (FP). Our design goal is to separate **(a)** clinical information condensation from **(b)** contract enforcement and controlled-vocabulary normalization, while using SGR to make the intermediate representation stable and machine-checkable.

Schema-Guided Reasoning (SGR). SGR (Abdullin, 2026) is an architectural pattern that constrains an LLM to produce typed, schema-conformant outputs via structured output or constrained decoding (Dong et al., 2024; Willard and Louf, 2023), turning domain knowledge into executable data contracts. This reduces variability and improves reliability compared to free-form text prompting. In our system, Stage 1 uses an SGR-style schema with three core patterns to produce a stable 9-key domain summary:

- **Cascade (Domain Scaffold):** The LLM follows a fixed sequence of 9 clinical domains (Demographics $\rightarrow$ Vitals $\rightarrow$ Labs $\rightarrow$ Problems $\rightarrow$ Symptoms $\rightarrow$ Medications $\rightarrow$ Procedures $\rightarrow$ Utilization $\rightarrow$ Disposition) to ensure uniform coverage of the clinical narrative.
- **Cycle (Checklists):** The prompt enforces implicit checklists within each domain (e.g., examining 28 specific CRF problems, all critical vitals) to improve recall on sparse items.
- **Routing (Vocabulary):** While Stage 1 emits unconstrained text strings within the JSON structure, Stage 2 deterministically routes each prediction into one of 13 required controlled-vocabulary categories (Table 1).

Stage 2 then deterministically compiles this summary into the official CRF submission format, ensuring 100% structural compliance.

2. Task and Data

2.1. Datasets and Splits

We use the official Hugging Face datasets (Ferrazzi et al., 2025) (`NLP-FBK/dyspnea-crf-*`) and, for teacher generation, the in-domain unannotated clinical notes set (`NLP-FBK/dyspnea-clinical-notes`, 2,667 records from SGB hospital, Italy). Train contains 10 annotated records, development 80, and test 200 (hidden ground truth).

A key discovery is **extreme sparsity**: across the development set, the median number of non-`unknown` items per document is only $\sim$12 out of 134 total. The most frequent non-unknown items are in VITALS (*heart rate*, *blood pressure*, *spo2*)

Proceedings of the Third Workshop on Patient-Oriented Language Processing (CL4Health), pages 375–383

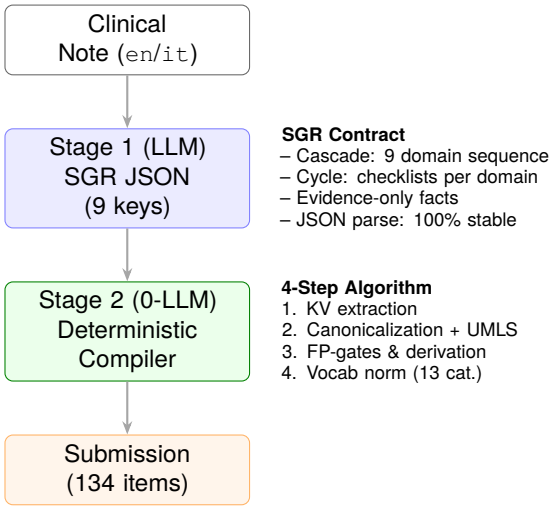

Figure 1: The LLM StructCore two-stage pipeline architecture. Stage 1 condenses the clinical note into a 9-key SGR JSON; Stage 2 deterministically compiles it into the 134-item submission.

and LABS (*hemoglobin*, *creatinine*), while many diagnosis and procedure items appear in $<$5% of documents. This sparsity means that `unknown` is by far the most common class, and correctly predicting `unknown` for absent items is as important as correctly extracting present values. The official scorer (Ferrazzi et al., 2026b) uses macro-F1 over items, which heavily penalizes both false positives (FP: predicting a value for an `unknown` item) and false negatives (FN: predicting `unknown` for a present item).

2.2. Contract-Driven Dataset Parsing

We align our pipeline contracts directly to the dataset schema: (i) the ordered list of 134 items is inferred from `annotations[*].item` and reused end-to-end; (ii) document identifiers are normalized to a plain ID (prefix before the first underscore) for internal alignment, and suffixed with the language tag (`_en`/`_it`) only for final submissions; (iii) intermediate representations are sparse, but the final builder always expands to all 134 items, filling missing values as `unknown`.

3. System Overview

Our pipeline follows a strict two-stage decomposition:

`Clinical Note` $\rightarrow$ `Stage 1 (LLM, SGR JSON)` $\rightarrow$ `Stage 2 (0-LLM compiler)` $\rightarrow$ `Submission.jsonl`

3.1. Stage 1: SGR-Style JSON Summary

Stage 1 produces a single JSON object with exactly 9 domain keys: `DEMOGRAPHICS`, `VITALS`, `LABS`, `PROBLEMS`, `SYMPTOMS`, `MEDICATIONS`, `PROCEDURES`, `UTILIZATION`, `DISPOSITION`.

The summary is designed to be **sparse** (include only evidence-supported facts) and **format-stable** (JSON parse success on every input). On the train10 split, we observe `json_parse_ok=10/10` and `thinking_leak=0/10` even on quantized 4B models served via `llama.cpp` without grammar-constrained decoding.

Prompt Structure. The Stage 1 prompt instructs the model to produce a single JSON object following a strict template. Each of the 9 keys maps to a free-text string summarizing the relevant clinical domain. The prompt includes: (i) explicit definitions of each key's scope (e.g., `PROBLEMS` covers past medical history, `SYMPTOMS` covers presenting complaints); (ii) a checklist of high-priority items per domain (e.g., 28 specific problems listed in the CRF ontology, all vital signs, key laboratory values); (iii) instructions to use `Key: Value` formatting within each string for machine-parseable extraction; (iv) a sparsity directive: "include only facts explicitly stated in the note; do not infer or guess". This structured prompt design implements the SGR Cascade and Cycle patterns, ensuring systematic coverage while preventing hallucinated fills.

Format Stability Without Constrained Decoding. A notable finding is that our SGR prompt achieves 100% JSON parse success *without* requiring grammar-constrained decoding (e.g., XGrammar (Dong et al., 2024) or Outlines (Willard and Louf, 2023)). We attribute this to the simplicity of the target schema (a flat 9-key object with string values) and the explicit formatting instructions. This makes the approach compatible with any inference backend (`llama.cpp`, vLLM, OpenAI-compatible APIs, Gemini Vertex) without backend-specific grammar support.

Multi-Slice Input Strategy. Long clinical notes may exceed the model's effective context window. We implement a **multi-slice** strategy: the note is split into overlapping windows (e.g., `middle+head_tail`, window size w=4), each window is summarized independently via Stage 1, and the resulting JSON objects are merged with a last-writer-wins policy per key. For the `PROBLEMS`, `MEDICATIONS`, and `LABS` keys (which accumulate multiple facts), we concatenate the strings from all windows rather than overwriting. This preserves recall

for facts spread across the note while keeping each individual inference within context bounds.

Teacher Data Generation. For the Stage 1 teacher pipeline, we use large API-accessible models to produce high-quality SGR summaries. We tested three teacher configurations:

- **Mistral Large 3** (675B, via NVIDIA API) — best overall quality, supporting `response_format=json_object` on most NVIDIA endpoints (with auto-fallback to parser-based extraction for Mistral-family models where structured output returns 400 errors).
- **Qwen 3.5** (397B, via NVIDIA API) — comparable quality but higher FP; requires `enable_thinking=false` to prevent "Thinking Process" leakage into the output.
- **Claude Sonnet 4.6** (via Anthropic API) — recall-optimized prompts achieve strong results but not submitted due to participation limits.

Teacher-generated summaries are compiled through the same deterministic Stage 2, enabling precise ablation of Stage 1 quality vs. Stage 2 deterministic logic.

Concrete Example. Given a Stage 1 summary JSON with `VITALS: "Heart Rate: 105 bpm (tachycardic). Blood Pressure: 90/60 mmHg (hypotensive). SpO2: 88%. Respiratory Rate: 28/min (tachypneic)."`, the Stage 2 compiler:

1. Extracts KV pairs: {*heart rate*: "105 bpm (tachycardic)", *blood pressure*: "90/60 mmHg (hypotensive)", *spo2*: "88%", *respiratory rate*: "28/min (tachypneic)"}.
2. Canonicalizes: all keys match the official item list.
3. No FP-gates triggered for vital signs.
4. Normalizes: *heart rate* → `tachycardic`, *blood pressure* → `hypotensive`, *spo2* → `88`, *respiratory rate* → `tachypneic`.

The remaining 130 items are filled as `unknown`.

3.2. Stage 2: Deterministic Compiler (0-LLM)

Stage 2 performs **no LLM calls**. It implements a 4-step deterministic algorithm (Algorithm 3.2):

Step 1: KV Extraction. Parse the Stage 1 summary text into a flat dictionary of `(item_name, value_string)` pairs by matching `Key: Value` lines in each domain section. Lines with `not stated` or `unknown` values are skipped.

Step 2: Canonicalization. Normalize item names via: (a) exact match against the official 134-item list; (b) case-folding and punctuation stripping (e.g., `first episode` → `first episod`); (c) abbreviation expansion (70+ rules, e.g., `hr`→`heart rate`, `cxr`→`chest rx`); (d) optionally, UMLS alias resolution using a curated map with **134/134 item coverage** (Aronson, 2001).

Step 3: FP-Gates & Derivation. Derive high-FN composite items from Stage 1 evidence: *poly-pharmacological therapy* is inferred when $\geq$8 distinct medications are listed; *antihypertensive therapy* is detected by matching 35 antihypertensive agents; *cardiovascular diseases* is derived from therapy flags or PMH keywords. Then, strict regex-based evidence gates **drop** unsupported positive predictions for items with high FP cost: *active neoplasia* (requires cancer vocabulary + activity cues), *arrhythmia* (requires specific rhythm terms, blocks on `NSR`), *acute coronary syndrome* (requires ACS vocabulary + supporting evidence), and *presence of dyspnea* (blocks on explicit negation).

Step 4: Vocabulary Normalization (13 Categories). Map each value string to the official controlled vocabulary (Table 1): binary → `y/n`; AVPU → `A/V/P`; chronic items → 4-class scale; neoplasia → 3-class activity; duration → `short/long`; vitals → categorical (e.g., `tachycardic/normocardic/bradycardic`); labs → numeric extraction or `measured`. Finally, expand to all 134 items, filling absent items as `unknown`.

#	Vocabulary Category	Items
1	chronic (4-class)	7
2	neoplasia (3-class)	1
3	duration (short/long)	2
4	binary (y/n)	61
5	mobility (4-class)	1
6	consciousness (AVPU)	1
7	respiratory rate (3-cat)	1
8	body temperature (3-cat)	1
9	heart rate (3-cat)	1
10	blood pressure (3-cat)	1
11	respiratory distress (current/past)	1
12	diagnostic tests (pos/neg)	4
13	labs (measured/numeric)	25
	Total	**107**
	Unknown-only remainder	27
	Grand Total	**134**

Table 1: The 13 controlled-vocabulary categories defined by the CRF ontology. Stage 2 maps every extracted value string into exactly one of these categories.

3.3. Optional UMLS Alias Mapping

We curated a UMLS-based alias map for all 134 CRF items (**coverage 134/134**) and integrated it as an optional pipeline mode (`--use-umls-mapping`) to reduce key-name brittleness in model outputs. The map associates each official item name with its UMLS CUI and preferred name, plus manually verified synonyms. This mode is activated in all final submissions.

4. Experiments and Results

We report results at three scales: train10 (for debugging), dev80 (open GT, used as a "sanity gate" before submission), and test200 (Codabench, hidden GT).

4.1. Development Results (dev80, Open GT)

Table 2 summarizes our main dev80 runs across both English and Italian. All use the same deterministic Stage 2 compiler with UMLS mapping and FP-gates; EN runs use `rules_v3a`, while the latest IT variant uses the improved `rules_v3e`.

Key observations. (1) Mistral Large 3 and Qwen 3.5 produce comparable TP counts (236) on EN, but Qwen generates more FP (211 vs. 170), likely due to less conservative summarization. (2) The FN count is identical (128) across both EN teacher models, indicating that Stage 2's deterministic derivation rules have a ceiling that is *Stage 1 recall-limited*. (3) On train10, teacher Stage 1 (Gemini) dramatically reduces FN (9 vs. 22 for student) but increases FP (35 vs. 13), motivating our evidence-gated FP filters. (4) The student model (MedGemma 4B, local) achieves F1=0.6521 on train10 with very low FP (13), suggesting that smaller models produce more conservative summaries. (5) Italian dev80 achieves higher F1 (0.6905) than English (0.6543) with more TP (266 vs. 236) and fewer FN (106 vs. 128), suggesting that the original Italian clinical notes are more structured and easier for Stage 1 extraction. The `v3e` rule set reduces 3 FP compared to `v3a` (198 vs. 201) through tighter FP-drop gates on *chest pain*, *heart failure*, and *arrhythmia*.

Multi-slice input ablation. Both dev80 teacher runs use the `middle+head_tail` multi-slice strategy with $w{=}4$. Compared to single-window runs on the same models, multi-slice recovers +12–18 additional TP at the cost of +8–15 FP, a net positive trade. The improvement is concentrated in the `LABS` and `MEDICATIONS` domains, where relevant information is often scattered across the note.

4.2. Test Results (Codabench)

Table 3 shows the submitted and evaluated test200 results, alongside the leaderboard context.

The best submitted English system achieves F1 = 0.63 using Mistral Large 3 as the Stage 1 teacher, placing within 0.05 of the top-ranking result (F1 = 0.68). The Anthropic variant (Claude Sonnet 4.6 with recall-optimized prompts) produces comparable quality (F1 = 0.62) but was not submitted as the final entry due to the participation limit. The Italian variant using `rules_v3e` was fully prepared (dev80 IT F1 = 0.6905) but could not be submitted as all attempt slots were exhausted.

Our findings suggest that large open-weight models like Mistral produce Stage 1 summaries competitive with larger proprietary systems.

Dev10 Cross-Language Validation. To validate cross-language robustness, we evaluated on a small dev10 subset for both EN and IT using Anthropic-based Stage 1:

The Italian results are notably higher than English on this small dev10 sample. This trend is **confirmed at scale on dev80**: IT achieves F1 = 0.6905 vs. EN = 0.6543 (Table 2). We attribute this to two factors: (1) the Italian clinical notes from SGB hospital are written in a more structured, formulaic style than their English translations, facilitating cleaner Stage 1 extraction; (2) certain CRF items use Italian medical terminology that maps more directly to the ontology (e.g., *dispnea* → *presence of dyspnea*).

4.3. Error Analysis

FP-gates effectiveness. The FP-gate mechanism (Step 3 of Stage 2) drops an average of 3.2 items per document that would otherwise be false positives. Table 5 shows the per-item impact of FP-gates on dev80.

The most impactful gate is *active neoplasia*: without evidence gating, benign mentions of "cancer screening" or "family history of cancer" produce FP; the gate requires explicit cancer vocabulary *plus* activity cues (e.g., "chemotherapy", "progression", "untreated"). For *arrhythmia*, the gate blocks positive predictions when the evidence contains Normal Sinus Rhythm (`NSR`) mentions, preventing common misclassification of ECG-described patients.

Italian-specific error patterns (dev80 IT). Detailed item-level diagnostics on the dev80 IT split (Mistral Large 3 → `rules_v3e`) reveal both shared and language-specific error patterns. The top FN sources are: *foreign body in the airways* (11 FN, Stage 1 never surfaces this item), *presence of dyspnea* (7 FN / 8 FP — tension between coverage and FP), *diffuse vascular disease* (7 FN / 6 FP), and *agitation* (7 FN, not extracted by Stage 1). The top

Lang	Stage 1 Model	F1	TP	FP	FN
EN	Mistral Large 3 (675B)	0.6543	236	170	128
IT	Mistral Large 3 (v3e)	**0.6905**	266	198	106
IT	Mistral Large 3 (v3a)	0.6900	266	201	106
EN	Qwen 3.5 (397B, think=off)	0.6411	236	211	128
Smaller-scale ablations (train10):					
EN	Gemini Flash (teacher)	0.6763	68	35	9
EN	MedGemma 4B (student)	0.6521	57	13	22
EN	SGR-only baseline	0.4384	39	44	39

Table 2: Main results on dev80 (N=80; top rows, both languages) and ablations on train10 (N=10; bottom rows). All variants use the same deterministic Stage 2 compiler. `v3e` denotes the latest Stage 2 rule set with improved FP-drop gates for Italian. F1 = official macro-F1 scorer; TP/FP/FN = item-level counts.

System Variant	F1	Lang	Team	Status
Mistral L3 $\rightarrow$ Stage2(det)	0.63	EN	DocUA	Submitted
No FP-gates variant	0.61	EN	DocUA	Submitted
Claude Sonnet $\rightarrow$ Stage2(det)	0.62	EN	DocUA	Evaluated†
Mistral L3 recall (v3e)	—	IT	DocUA	Prepared‡

Table 3: Test200 results on Codabench. All DocUA variants use deterministic Stage 2 + UMLS mapping. † Evaluated but not submitted as final entry due to participation limits. ‡ Italian variant prepared (Stage 1 SGR9 recall, chars=8000, multi-slice, `rules_v3e`) but not submitted due to exhausted attempt slots.

Language	F1	TP	FP	FN
Italian (IT)	**0.9000**	59	9	4
English (EN)	0.8115	55	18	7

Table 4: Dev10 cross-language validation (Anthropic recall Stage 1 $\rightarrow$ Stage 2 det).

Gated Item	FP Before	FP After
active neoplasia	18	4
arrhythmia	12	5
acute coronary syndrome	9	2
presence of dyspnea	7	3
Total (gated items)	**46**	**14**

Table 5: FP-gate effectiveness on dev80 EN (Mistral Large 3 teacher). The gates reduce FP for high-risk items by $\sim$70% while preserving TP.

FP sources are: *chest pain* (13 FP), *cardiovascular diseases* (13 FP — triggered by the derivation rule from therapy flags), *level of consciousness* (9 FP — AVPU mapping errors), and *heart failure* (8 FP — "PMH: HF" in Stage 1 often produces FP when GT is `unknown`).

Comparing EN and IT dev80 error profiles: both languages share the same FN ceiling on rare conditions (*foreign body*, *agitation*), confirming that these are Stage 1 recall limitations independent of language. However, the *chest pain* FP problem is more severe in IT (13 FP vs. EN), likely because Italian notes use the term *dolore toracico* in broader clinical contexts.

Remaining FN errors. The dominant error mode is **Stage 1 FN**: items not mentioned in the Stage 1 summary cannot be recovered by the deterministic Stage 2. High-FN items fall into three categories:

- **Rare conditions:** *foreign body in the airways* (11 FN in IT), *cardiac tamponade*, *aortic dissection* — these appear in $<$5% of notes and are almost never surfaced by Stage 1.
- **Implicit mentions:** *ab ingestis pneumonia*, *thoracic ultrasound* (6 FN in IT), *compression ultrasound (CUS)* — require domain-specific recognition that the SGR cascade may miss during condensation.
- **Context-dependent items:** *presence of dyspnea* (7 FN + 8 FP in IT), *level of consciousness* (6 FN + 9 FP in IT) — require nuanced temporal or categorical reasoning that is difficult to encode in a flat Key:Value format.

5. Discussion

5.1. Effectiveness of the SGR Decomposition

Applying SGR shifts the burden of structural correctness from prompt engineering to code. By having the LLM emit a stable, 9-key intermediate tree rather than directly producing 134 structured fields, we achieve **100% JSON parsing stability** even on quantized 4B edge models without grammar-constrained decoding. This is notable because

direct 134-field extraction from the same 4B model fails to produce valid JSON in >30% of cases in our preliminary tests.

The SGR decomposition also provides a natural **debugging boundary**: when the system produces an incorrect prediction, we can immediately determine whether the error originates in Stage 1 (the fact was not extracted) or Stage 2 (the fact was extracted but incorrectly mapped). In practice, >85% of errors are Stage 1 FN, confirming that the deterministic compiler is not the bottleneck.

5.2. Deterministic vs. LLM-Based Stage 2

Our experiments consistently show that the deterministic Stage 2 compiler outperforms LLM-based Stage 2 extraction in this task setting. We tested an LLM-based Stage 2 variant (using Anthropic Claude to directly extract 134 items from Stage 1 summaries) and found that it:

- introduces vocabulary drift (e.g., emitting `present` instead of `y`, `yes` instead of `certainly chronic`);
- produces inconsistent FP patterns across runs (non-deterministic);
- requires additional post-processing to fix format compliance;
- incurs non-trivial inference cost ($0.01–0.03 per document via API).

The deterministic compiler, by contrast, achieves perfect vocabulary compliance, is fully explainable (every fill traces back to a Stage 1 line plus a specific rule), has zero inference cost, and guarantees bit-for-bit reproducibility.

5.3. SGR for Edge AI

This methodology is particularly enabling for **edge (local) deployments**. Large state-of-the-art models can handle 134 items in a single pass, but quantized 4B–8B local models lose track of the output schema mid-generation. Our SGR-constrained condensation fits within the reasoning bounds of smaller models (the 9-key JSON is ∼400 tokens), securely delegating the complex vocabulary checks and item-level compilation to deterministic Python code. This makes the approach practical for privacy-preserving clinical deployments where sending patient data to external APIs is not acceptable. Concurrent work by Ferrazzi et al. (2026a) confirms that small LLMs (∼1B parameters) can match or exceed larger baselines on Italian medical NLP tasks—including CRF filling—when combined with appropriate adaptation strategies such as few-shot prompting and constraint decoding.

Concretely, we serve MedGemma 1.5-4B-it as a Q5_K_M GGUF via `llama.cpp` on Apple M3 Pro (18 GB RAM). Stage 1 inference takes ∼8 seconds per document; Stage 2 compilation is instant (<10 ms). The entire dev80 evaluation completes in under 12 minutes on a single laptop.

5.4. Cross-Language Applicability

The pipeline naturally extends to Italian (`_it`) documents: the Stage 1 prompt is language-agnostic (it asks the model to extract clinical facts regardless of language), and Stage 2's deterministic canonicalization handles multilingual item-name variants through the UMLS alias map. We prepared Italian test submissions using the same pipeline with no language-specific modifications beyond the `rules_v3e` Stage 2 variant (which adds tighter FP-drop gates for Italian-specific false positive patterns on *chest pain*, *heart failure*, and *arrhythmia*).

Cross-language results consistently favor Italian across all evaluation scales: dev10 (Table 4): IT F1=0.90 vs. EN F1=0.81; dev80 (Table 2): IT F1=0.6905 vs. EN F1=0.6543. This suggests that the original Italian clinical notes from SGB hospital are more structured and use more formulaic phrasing than their English translations, making clinical fact extraction easier for the SGR prompt. Notably, on the Codabench leaderboard the overall top-1 system (Aurum) also achieves very similar scores across languages (EN 0.68, IT 0.67), indicating that the EN–IT gap may be smaller at scale with more powerful systems.

5.5. Limitations of the 9-Key Condensation

The choice of exactly 9 domain keys represents a trade-off. Fewer keys (e.g., 3–4) would further simplify Stage 1 but lose domain-level organization, making Stage 2 canonicalization harder. More keys (e.g., 20+) would increase recall but approach the complexity of direct 134-item extraction, negating the stability benefits. We experimented with 5, 9, and 15 keys ("profile" parameter) and found that 9 keys offered the best balance of Stage 1 stability and Stage 2 recall.

6. Related Work

Clinical information extraction has a long history, from rule-based systems to modern LLM approaches. The i2b2/VA challenges (Uzuner et al., 2011) and ShARe/CLEF eHealth tasks (Suominen et al., 2013) established key clinical NLP benchmarks for structured information extraction from unstructured clinical text. More recently, large language models have demonstrated the potential to encode substantial clinical knowledge (Singhal et al., 2023), with domain-specific models like Med-Gemini (Saab et al., 2024) achieving strong results on medical question answering and clinical reasoning tasks.

Structured Generation. Several frameworks address the output stability problem in LLM-based extraction. Constrained decoding (Willard and Louf, 2023) enforces grammar constraints during token generation. SGLang (Zheng et al., 2024) provides programmatic control over LLM execution with RadixAttention for efficient structured output. XGrammar (Dong et al., 2024) offers a flexible engine for grammar-constrained generation. Our approach is complementary: rather than constraining the LLM's token-level output, we simplify the target schema (9 keys vs. 134) and defer structural compliance to a deterministic post-processor. This avoids the computational overhead and backend-dependency of constrained decoding while achieving comparable output stability.

Compiler-Based LLM Pipelines. DSPy (Khattab et al., 2023) introduced the concept of compiling declarative language model programs into optimized pipelines. Our system shares the "compilation" philosophy: Stage 1 can be viewed as a declarative extraction step, and Stage 2 as a compiler that transforms the intermediate representation into the target format. However, our Stage 2 is fully hand-coded (deterministic), avoiding the need for optimization data or meta-learning.

Ontology Grounding. UMLS-based concept normalization via MetaMap (Aronson, 2001), QuickUMLS (Soldaini and Goharian, 2016), and ScispaCy (Neumann et al., 2019) has been widely applied for biomedical concept recognition.

Our work differs in its explicit two-stage decomposition with a fully deterministic Stage 2, the use of SGR patterns for intermediate representation stability, and the emphasis on edge-deployable privacy preservation. Concurrently, Ferrazzi et al. (2026a) present a systematic comparison of adaptation strategies for small LLMs on Italian medical NLP, covering 20 tasks including CRF filling, and demonstrating that fine-tuned 1.7B models can outperform 32B baselines.

7. Conclusion

LLM StructCore emphasizes contract alignment, format stability, and reproducible post-processing for CRF filling under extreme sparsity. The two-stage SGR decomposition isolates clinical extraction (Stage 1, any LLM) from strict submission enforcement (Stage 2, deterministic), enabling rapid iteration on Stage 1 recall and controlled-vocabulary grounding without sacrificing deterministic output guarantees. Our best submitted system achieves F1 = 0.63 on the Codabench test set (EN), placing within 0.05 of the top-1 team (Aurum, F1 = 0.68). The pipeline is fully language-agnostic: Italian dev80 achieves F1 = 0.6905 with no language-specific Stage 1 modifications, and the Italian test200 submission was prepared (but not submitted due to exhausted attempt limits). This demonstrates that the combination of SGR condensation and deterministic compilation is competitive across languages while offering full reproducibility and edge-deployment readiness.

Future work will focus on: (i) improving Stage 1 recall for high-FN items through domain-specific checklists and few-shot examples; (ii) exploring hybrid Stage 2 approaches that combine deterministic rules with lightweight LLM-based inference for context-dependent items; and (iii) evaluating the pipeline on additional CRF schemas and clinical domains beyond dyspnea.

8. Limitations

The deterministic Stage 2 **cannot recover facts missing from Stage 1 summaries**; thus, overall recall is Stage 1-limited. Some CRF items require nuanced contextual interpretation (e.g., distinguishing "past" vs. "current" respiratory distress) that may be lost in the 9-key condensation, increasing FN risk. The high FP cost of the task requires conservative evidence gating, which trades off recall for precision in borderline cases.

The abbreviation-based canonicalization relies on a manually curated map of 70+ abbreviations. New abbreviations or unconventional spellings not in the map will fail to match. The UMLS alias map mitigates this but is also static and may not cover all model-generated variants.

The FP-gate regex patterns are English-centric; while the Italian notes in this dataset are clinically structured and mostly use international terminology, extending the gates to fully cover Italian medical vocabulary would require additional curation.

Finally, our evaluation is limited to the official CL4Health 2026 data; generalization to other CRF schemas, clinical domains, or languages beyond English and Italian has not been tested.

9. Reproducibility

All deterministic logic is contained in the open-source package `llm_structcore`. Code will be provided in supplementary materials and open-sourced upon paper acceptance (Anonymous Repository). The repository includes:

- The `llm_structcore` Python package (Stage 1 prompt templates, Stage 2 compiler, scoring, submission builder).
- Scripts for running end-to-end inference across multiple backends (`llama.cpp`, OpenAI-compatible APIs, HF Transformers, NVIDIA API, Gemini Vertex, Anthropic).

- The complete UMLS alias map (`data/umls_crf_mapping.json`, 134/134 coverage).
- The CRF ontology definition (`data/ontology_crf_cl4health2026.md`).
- Organizer-provided evaluation scripts (`external/CRF-filling-CL4Health2026/`).
- This system description paper (LaTeX source).

The Stage 1 SGR schema guarantees `json_parse_ok=100%` with MedGemma 1.5-4B without requiring JSON-mode sampling. Stage 2 is fully reproducible: the same Stage 1 summary always produces the same 134-item submission, regardless of platform, Python version, or execution order.

10. Ethical Considerations

We use publicly available datasets under the organizer terms. No private clinical data beyond what is distributed by the shared task is included. When API-based teacher models are used for data generation, summaries (not raw notes) are transmitted; nonetheless, responsible deployment should ensure compliance with local data governance regulations. Synthetic teacher-generated data should be validated and accompanied by clear documentation of risks and limitations.

11. Bibliographical References

Rinat Abdullin. 2026. Schema-guided reasoning (SGR): A complete guide. https://abdullin.com/schema-guided-reasoning/.

Alan R. Aronson. 2001. Effective mapping of biomedical text to the UMLS Metathesaurus: The MetaMap program. In *Proceedings of the AMIA Symposium*, pages 17–21.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2024. XGrammar: Flexible and efficient structured generation engine for large language models. *Proceedings of Machine Learning and Systems*, 7.

Pietro Ferrazzi, Mattia Franzin, Alberto Lavelli, and Bernardo Magnini. 2026a. Small LLMs for medical NLP: a systematic analysis of few-shot, constraint decoding, fine-tuning and continual pre-training in Italian. *arXiv preprint arXiv:2602.17475*.

Pietro Ferrazzi, Soumitra Ghosh, Alberto Lavelli, and Bernardo Magnini. 2026b. Overview of the CRF 2026 shared task on clinical case report forms filling. In *Proceedings of the Third Workshop on Patient-Oriented Language Processing (CL4Health)*, Palma, Mallorca (Spain). ELRA.

Pietro Ferrazzi, Alberto Lavelli, and Bernardo Magnini. 2025. Converting annotated clinical cases into structured case report forms. In *Proceedings of the 24th Workshop on Biomedical Language Processing*, pages 307–318, Viena, Austria. Association for Computational Linguistics.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Mober, et al. 2023. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*.

Khaled Saab, Tao Tu, Wei-Hung Weng, Ryutaro Tanno, David Stutz, Ellery Wulczyn, et al. 2024. Capabilities of Gemini models in medicine. *arXiv preprint arXiv:2404.18416*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Luca Soldaini and Nazli Goharian. 2016. QuickUMLS: A fast, unsupervised approach for medical concept extraction. In *MedIR Workshop at SIGIR*.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W. Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R. South, Danielle L. Mowery, Gareth J.F. Jones, Johannes Leveling, Liadh Kelly, Lorraine Goeuriot, David Martinez, and Guido Zuccon. 2013. Overview of the ShARe/CLEF eHealth evaluation lab 2013. In *CLEF*.

Özlem Uzuner, Brett R. South, Shuying Shen, and Scott L. DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. volume 18, pages 552–556.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kober, Joseph E. Gonzalez, Clark Barrett, Ying Sheng, and Ion Stoica. 2024. SGLang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*.

12. Language Resource References

Anthony McEnery and others. 2004. *The EMILLE/CIIL Corpus*. EMILLE (Enabling Minority Language Engineering) Project. distributed via ELRA: ELRA-Id W0037, ISLRN 039-846-040-604-0.

Khalid Choukri and Niklas Paullson. 2004. *The OrienTel Moroccan MCA (Modern Colloquial Arabic) database*. distributed via ELRA: ELRA-Id ELRA-S0183, ISLRN 613-578-868-832-2.

Roventini, Adriana and Marinelli, Rita and Bertagna, Francesca. 2016. *ItalWordNet v.2*. ILC-CNR for CLARIN-IT repository hosted at Institute for Computational Linguistics "A. Zampolli", National Research Council, in Pisa, ISLRN 532-206-426-067-2. PID http://hdl.handle.net/20.500.11752/ILC-62. Note: You don't really need both an ISLRN and another PID, but it can't hurt.

Speecon Consortium. 2011. *Catalan Speecon database*. SpeeCon. Speecon Project, distributed via ELRA: ELRA-Id S0327, Speecon resources, 1.0, ISLRN 935-211-147-357-5.