MedSNIP: Building and Benchmarking Snippet-Level Granularity for Medical Fact Verification

Anonymous ACL submission

Abstract

A medical claim's correctness often depends not on the claim alone, but on the clinical structure around it. A claim may require a lab reference range, a causal or conditional link, or patient-specific details to be judged correctly, and standard atom-level decomposition can fragment these dependencies, leaving the verifier with claims that are formally standalone but clinically incomplete. We reformulate medical fact-checking around *snippet-level* verification, where clause-grouped units preserve local clinical structure. We introduce MEDSNIP-BENCH, a human-annotated benchmark for snippet-level medical fact verification, and MEDSNIP, an automatic snippet-generation pipeline. MEDSNIP-BENCH covers 276 consumer-health and clinical-vignette responses, segmented into 2,524 snippets with dual in-general and in-patient-context labels and six structural pattern codes. MEDSNIP is evaluated against human snippet boundaries on MEDSNIP-BENCH and then used to generate snippet-level units for external corpora. Across MEDSNIP-BENCH, HEALTHFC, and MEDHALLU, snippet-level verification improves or preserves false-class F1, with the largest merge-pattern gain on causal-conditional clinical chains. It also reduces verification cost by 22–47%. We release MEDSNIP-BENCH, MEDSNIP, and our code at http://anonymous.for.review.

1 Introduction

Large language models (LLMs) are increasingly used to answer medical questions from patients, consumers, and clinicians (Singhal et al., 2023, 2025; Manes et al., 2024). A fluent but unsupported recommendation can shape patient decisions, mislead downstream systems, and propagate into clinical reasoning (Zhu et al., 2025). Automated fact-checking is a safety requirement for medical LLMs, not an evaluation convenience.

Currently, most fact-checking pipelines descend from FACTSCORE (Min et al., 2023) and its successors. They split a long-form generation into minimal standalone *atomic* claims, verify each claim independently, and aggregate the verdicts (Chern et al., 2023; Wei et al., 2024; Dhuliawala et al., 2024; Iqbal et al., 2024). This atom-level design is natural when each claim can be judged on its own. Medical answers often violate this assumption. A lab interpretation may require both a measured value and a reference range. A diagnosis may require a pattern of findings rather than any single finding in isolation. A recommendation may be true in general but wrong for the patient described in the question. In such cases, atomization can turn a clinically meaningful judgment into fragments that lack the information needed for verification. We instead use a coarser verification unit, called a *snippet*.

A snippet is a clinically meaningful group of related clauses, smaller than a full answer but coarser than an atom. It keeps together the information needed to verify one clinical judgment, such as a finding with its interpretation, a lab value with its reference range, or a diagnostic conclusion with its caveat. Figure 1 illustrates this on a clinical-vignette answer, where one judgment fragments into three or four atoms. This motivates treating granularity as a core design choice in medical fact-checking, not as pre-processing.

Motivated by these observations, we reformulate medical fact-checking around *snippet-level verification*. Rather than decomposing responses into atomic claims or evaluating entire answers holistically, we group semantically and clinically related clauses into coherent snippets that preserve the context needed to verify clinical judgments. Each snippet is annotated for both general medical correctness and patient-specific correctness, capturing errors that may be medically plausible in general but unsafe for the particular patient context.

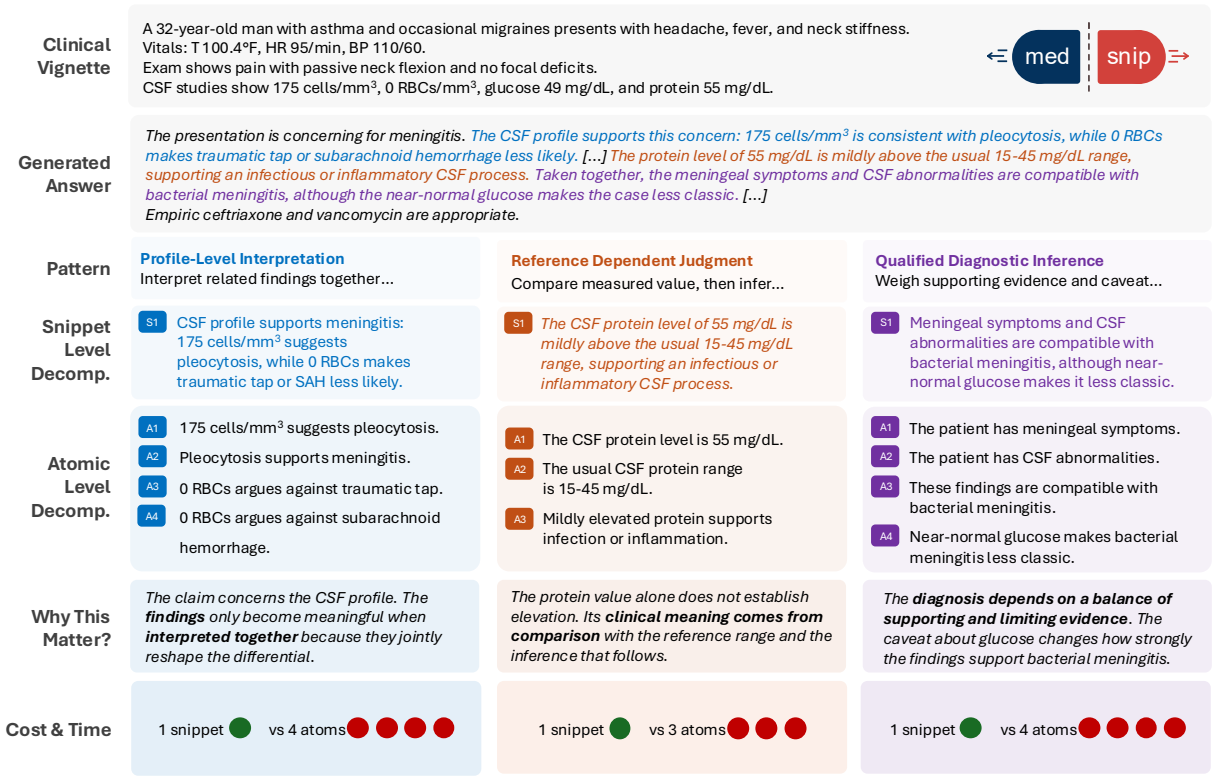


Figure 1: Motivating example from MEDSNIP-BENCH showing how atomization can split clinically coherent judgments. Each column presents one judgment from a clinical-vignette answer. Snippet-level decomposition preserves the linked evidence, interpretation, and caveat as one verifiable unit, while atom-level decomposition fragments the same judgment into three or four separate claims.

Existing medical fact-checking resources do not evaluate this formulation. HEALTHFC provides expert-labeled consumer-health claims and MEDHALLU provides a PUBMEDQA-derived hallucination benchmark, but both evaluate near-atomic claims (Vladika et al., 2024; Pandit et al., 2025). Domain-tailored decomposers and verifier diagnostics similarly remain within the atom-level paradigm (Huang et al., 2025; He et al., 2025). To fill this gap, we introduce **MEDSNIP-BENCH**, a human-annotated benchmark for snippet-level medical fact-checking, together with **MEDSNIP**, an automatic snippet-generation pipeline. Together, this paper contributes:

- A reformulation of medical fact-checking around snippet-level verification, where related clauses are checked as clinically coherent units.
- MEDSNIP-BENCH, with 276 consumer-health and clinical-vignette responses, 2,524 human-annotated snippet boundaries, dual *in-general* and *in-patient-context* labels, and six structural pattern codes.
- MEDSNIP, an automatic snippet-generation pipeline evaluated against human snippet boundaries on MEDSNIP-BENCH and used to generate snippet-level units for external corpora such as HEALTHFC and MEDHALLU.
- A cross-model, cross-dataset evaluation with Bias-Corrected and Accelerated (BCa) bootstrap confidence intervals, plus per-pattern analysis showing where snippet-level verification helps.

2 Related Work

Atomic fact-checking: FACTSCORE established a common paradigm for long-form factuality evaluation, where a generation is decomposed into atomic claims and each claim is checked against evidence (Min et al., 2023). Subsequent systems adopt claim-level verification pipelines (Chern et al., 2023; Wei et al., 2024; Gao et al., 2023; Iqbal et al., 2024). Recent work questions whether the smallest claim is always the best verification unit. Wanner et al. (2024) show that verifier accuracy depends on decomposition granularity, while Lu et al. (2025) learn a verifier-preferred atomicity policy that improves over default atomization. Rather than

learning an instance-specific policy, we show that a coarser, clause-grouped unit is sufficient, cheaper, and aligned with medical reasoning.

Medical fact-checking: Medical factuality evaluation spans consumer-health and clinical question-answering settings. HEALTHFC provides expert-labeled consumer-health claims with associated abstracts (Vladika et al., 2024). MEDHALLU extends PUBMEDQA into a hallucination-detection benchmark with paired ground-truth and adversarially perturbed answers (Pandit et al., 2025). MEDSCORE adapts FACTSCORE-style decomposition to free-form medical answers and shows that a domain-tailored atomizer yields more valid claims than a general atomizer (Huang et al., 2025). MEDFACT documents an over-criticism failure mode, where stronger reasoning verifiers incorrectly mark correct medical text as false (He et al., 2025). Kim et al. (2025) conduct a large-scale expert evaluation of retrieval-augmented medical QA, showing that retrieval covers only a minority of clinician-identified must-have information and that standard RAG can degrade factuality. None of these resources release snippet-level structure or dual in-general / in-patient-context labels. In contrast, we introduce MEDSNIP-BENCH, sourced from a subset of the query corpus introduced by Kim et al. (2025), as the first human-annotated snippet-level medical fact-checking dataset that preserves clinically meaningful claims together with their surrounding contextual information. Our results highlight the need for snippet-style verification units in factuality benchmarks, rather than relying exclusively on atomized claims.

3 MEDSNIP-BENCH

MEDSNIP-BENCH is built on the benchmark released by Kim et al. (2025), which pairs 100 consumer-health queries from K-QA (Manes et al., 2024) with 100 USMLE-style clinical-vignette questions from MEDBULLETS (Chen et al., 2025) and carries expert true-or-false labels on individual atomic claims. We use **276 entries** containing **5,755 expert-labeled atomic claims**, partitioned into 140 consumer-health and 136 clinical-vignette entries using a word-length threshold on the original query. The parent corpus is naturally bimodal in query length, so the partition is unambiguous (see Appendix A.1). The atom-level labels are inherited from Kim et al. (2025), while MEDSNIP-BENCH contributes all annotations above the atom layer.

Snippets and Pattern Taxonomy: The atomic-claim layer is often too fine-grained for medical content, because many clinical judgments depend on information distributed across related clauses. To preserve these dependencies, we re-segment the corpus into *snippets*, clause-grouped verifiable units designed to retain shared entities, clinical framing, and causal or conditional relationships. Analysis across the parent benchmark showed that snippet boundaries consistently recur around six structural patterns. We identified these patterns through error analysis of an atomic-claim verification baseline, where recurring verification failures clustered into the same structural cases.

Patterns A–C, illustrated in Figure 1, are merge patterns, where multiple atoms should be verified together as a single snippet, while Patterns D–F are keep-atomic patterns, where the atom remains the appropriate verification unit. The six structural patterns are summarized below:

- **Pattern A** for enumerations of properties of a single subject (features, causes, or factors that only make sense as a set)
- **Pattern B** for causal or conditional chains whose links share evidence (clinical reasoning, including diagnostic logic, drug-effect chains, and threshold-dependent recommendations)
- **Pattern C** for conclusions together with the premises that warrant them (a diagnosis or recommendation whose verifiability depends on the supporting findings just stated).
- **Pattern D** for self-contained standalone facts or lab interpretations whose verifiability does not depend on surrounding clauses
- **Pattern E** for genuine topic shifts where merging would conflate distinct subjects
- **Pattern F** for distinct facts about the same subject that do not jointly support a single judgment, including isolated false atoms surrounded by true ones (a case where atomization protects the false claim from being absorbed into a true context).

Dual correctness labels: Each snippet receives two correctness judgments: *(i).* **in-general** indicates whether the snippet is correct as a general medical statement. *(ii).* **with-patient-context** indicates whether it is correct for the specific patient or query. This dual scheme captures cases that are generally true but contextually wrong, such as

Subset / Split	Entries	Claims	%F in-general	%F in-context	A	B	C	D	E	F
Parent corpus	276	5,755 atoms	–	–	–	–	–	–	–	–
Consumer	140	1,091 snippets	12.4	9.2	634	101	79	194	10	73
Vignette	136	1,433 snippets	14.6	11.4	374	124	242	423	29	241
Train	180	1,599 snippets	13.6	10.5	640	143	209	385	26	196
Dev	51	494 snippets	13.8	10.7	187	44	58	126	10	69
Test	45	431 snippets	13.7	10.0	181	38	54	106	3	49
Total	276	2,524 snippets	13.6	10.5	1008	225	321	617	39	314

Table 1: MEDSNIP-BENCH statistics by source subset and split, with the filtered parent corpus shown for reference. **%F in-general** and **%F in-context** are the percentage of snippets labeled FALSE under the in-general and in-patient-context labels. Pattern columns show counts for the six structural pattern codes.

recommendations that are reasonable in general medicine but inappropriate for the vignette at hand. In MEDSNIP-BENCH, 124 snippets, or 4.9%, have divergent in-general and with-context labels. These cases are concentrated in clinical-vignette entries, consistent with contextual error being especially relevant in patient-specific settings. Full definitions and worked anchor examples are in Appendix E.

3.1 Annotation

Six annotators with medical or biomedical-NLP backgrounds segmented and labeled the data using a shared interface (Figure 21) and written guidelines (see Appendix E). Snippet boundaries and pattern codes were treated as linguistic and logical judgments over the answer text, not as new clinical diagnoses. Annotators also saw the parent corpus's physician true-or-false labels for each atomic claim, which anchored the labeling task in expert annotations. For each entry, an annotator first identified the shared context (see Appendix A.3), such as patient demographics or query topic. They then applied the A–F pattern taxonomy to decide which atomic claims should be merged into the same snippet and which should remain separate. After grouping the claims, the annotator revised each snippet into a self-contained verifiable statement and assigned the two binary correctness labels.

On a stratified IAA set covering both source subsets, label outcomes, and merge patterns, agreement is substantial for the *in-general* label and moderate for the harder *with-patient-context* label, with Fleiss' κ of 0.662 and 0.472, respectively. Snippet boundary agreement is strong, with mean pairwise $ARI = 0.723$. Details are in Appendix A.4.

3.2 Dataset Statistics

After snippet annotation, the 5,755 atomic claims of the parent corpus map to **2,524 snippets**. Among these 2,524 snippets, 1,554 are multi-atom and 970 are single-atom, with a median of 2 atoms per snippet. The dataset is skewed toward true snippets so we use F_1^F, defined as F1 on false claims, as the primary evaluation metric. Table 1 summarizes the parent corpus, source subsets, splits, and structural pattern counts.

	MEDSNIP-B	HEALTHFC	MEDHALLU
Answers	276	750	1,000
Units	2,524	750	2,000
Unit type	snippet	atom	atom
Labels	dual	–	–
Patterns	A–F	–	–

Table 2: Comparison of MEDSNIP-BENCH with the two external datasets. Only MEDSNIP-BENCH has snippet units, dual context labels, and A–F pattern annotations.

The train, dev, and test partitions are entry-level splits that preserve source-subset proportions and snippet-level false rates across label dimensions (Appendix A.2). All 276 entries are used for the headline zero-shot snippet-vs-atom comparison. The splits support the retrieval-augmented verifier, which was developed on dev and confirmed on test. Consumer-health entries skew toward enumeration, while clinical vignettes contain more reasoning chains, diagnostic conclusions, and standalone clinical interpretations.

We use two external medical fact-checking datasets, HEALTHFC (Vladika et al., 2024) and MEDHALLU (Pandit et al., 2025), to test whether the snippet-vs-atom comparison generalizes beyond MEDSNIP-BENCH. From HEALTHFC, we use the 327 claims with binary support labels and exclude the 423 insufficient-evidence claims from F1 computation. From MEDHALLU, we use 2,000 binary items formed from 1,000 paired ground-

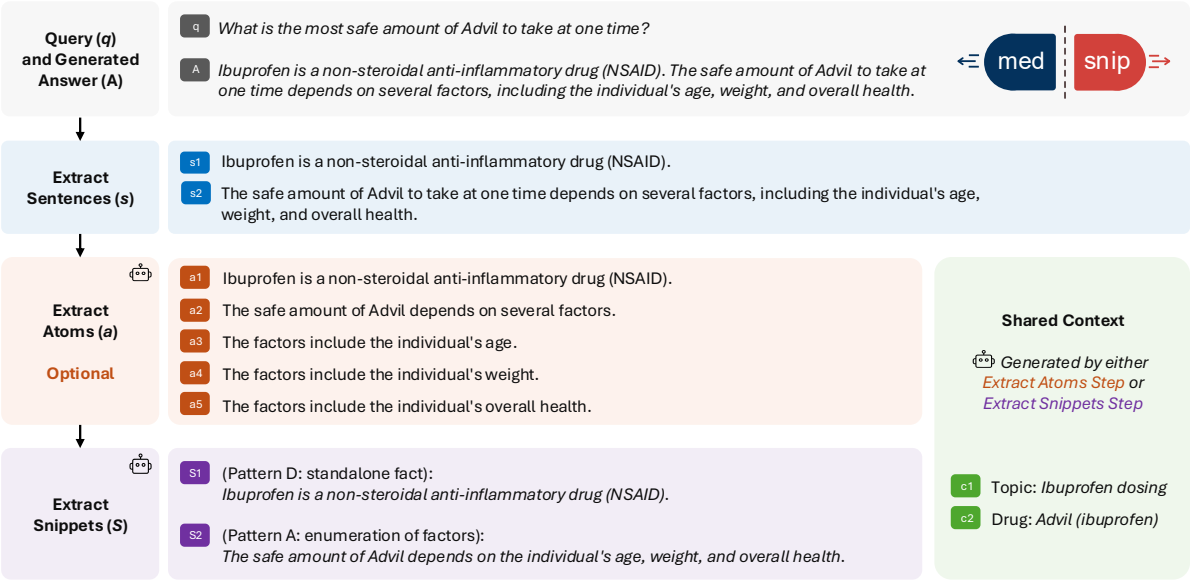


Figure 2: MEDSNIP snippet-generation pipeline on a worked Advil example. After deterministic sentence splitting, Mode 1 extracts atoms and shared context, then clusters atoms into snippets. Mode 2 generates snippets directly from sentences. Both modes return snippets with source pointers, A–F pattern codes, and shared context.

truth and hallucinated answers derived from PUBMEDQA (Jin et al., 2019). Table 2 summarizes the three corpora. Since neither external dataset provides snippet boundaries or dual context labels, we apply the MEDSNIP pipeline to produce snippets on those corpora before evaluation.

4 Method

We compare medical fact-checking pipelines along three axes, *(i)*. the unit sent to the verifier, *(ii)*. the context shown during verification, and *(iii)*. the verifier design. The core comparison is between atom-level and snippet-level verification.

Given a long-form medical answer A, a decomposer maps A into verifiable units $\{u_1, \ldots, u_k\}$. A verifier assigns each unit a label in $\{\text{TRUE}, \text{FALSE}\}$.

We compare two decomposition granularities. The **atom-level** condition follows the FACTSCORE lineage and verifies the smallest standalone factual statements. The **snippet-level** condition verifies the clause-grouped units. For MEDSNIP-BENCH, snippet boundaries come from human annotation. For HEALTHFC and MEDHALLU, snippets are generated automatically with the MEDSNIP pipeline.

4.1 MEDSNIP Pipeline

MEDSNIP pipeline converts a long-form answer A and optional query q into snippets S. Each snippet includes source pointers, one of the A–F structural pattern codes, and shared context ctx. Source pointers identify the atoms or sentences that support the snippet. Shared context is a short structured summary of the answer's topic, entities, and patient or query framing. The LLM steps use this context to interpret otherwise underspecified units.

As a first step, we apply deterministic sentence splitting after removing citation markers, bullets, numbering, and headers. To compare automatic snippets with human snippets, we align source atoms to sentences using normalized-token overlap and substring matching, enabling sentence-set F1 evaluation without an additional model.

The pipeline has two modes **Mode 1: Atomize-then-group** first extracts atoms and then uses an LLM to cluster them into snippets, mirroring the benchmark construction process. **Mode 2: Snippet-direct** skips the atom intermediate and generates snippets directly from sentences. Figure 2 summarizes the two modes on a worked Advil example, where the factors enumerated in the answer are either extracted as atoms and grouped, or grouped directly into snippets.

The two modes trade compute against fidelity to the annotation workflow. Atomize-then-group uses two LLM calls and mirrors human annotation, while snippet-direct uses one LLM call. We use GPT-5.4 (OpenAI, 2026) as the decomposer across modes and datasets, so atom and snippet baselines are not confounded by decomposition models. Full prompts are in Appendix B.

We compare the two modes against human snippets on the full MEDSNIP-BENCH. Atomize-then-group better matches human snippets than snippet-direct, with higher sentence-level F1 (0.756 vs. 0.734) and higher mean embedding similarity (0.824 vs. 0.794). We therefore use atomize-then-group as the default snippet construction procedure and report snippet-direct as a one-call ablation. The full pipeline-quality evaluation is in Appendix A.5. The pattern codes are retained for analysis and inspection, while the main per-pattern results use the human pattern annotations in MEDSNIP-BENCH.

4.2 Verification Modes

For each unit u, we evaluate two verification modes. *(i)* **CLAIM-ONLY verification** gives the verifier only u, testing whether the unit is self-contained. *(ii)* **FULL-CONTEXT verification** gives the verifier u with the original question and full generated answer. This context is used only to disambiguate the unit, recover omitted entities, and resolve references; the verifier is instructed not to treat the generated answer as evidence. This tests whether context lost during decomposition can be recovered at verification time.

4.3 Retrieval-Augmented Verifier

Single-call verifiers judge each unit from parametric knowledge alone. This can be brittle for medical text, where verification may depend on drug indications, doses, reference ranges, guidelines, or external evidence. We therefore also evaluate a retrieval-augmented verifier. It searches for evidence over multiple rounds, reasons over the accumulated evidence, and returns TRUE, FALSE, or ABSTAIN when the evidence is insufficient.

The verifier maintains an evidence pool $\mathcal{E}_t$, initialized as

$$\mathcal{E}_0 = \emptyset$$

At retrieval step t, the model writes a search query from the unit, original query, and current evidence:

$$s_t = \mathrm{LLM}_{query}(u, q, \mathcal{E}_{t-1})$$

Retrieval runs in parallel over Google Serper[1] for general web evidence and the PubMed E-utilities API[2] for biomedical abstracts. Each backend returns up to $k = 3$ passages per query, and the evidence pool accumulates passages across rounds:

$$\mathcal{E}_t = \mathcal{E}_{t-1} \cup \mathrm{Ret}_{web}(s_t) \cup \mathrm{Ret}_{biomed}(s_t)$$

[1] https://serper.dev/
[2] https://www.ncbi.nlm.nih.gov/books/NBK25501/

The verifier then reads the accumulated evidence and predicts a label with a self-reported confidence score c_t in $[0, 1]$:

$$(\hat{y}_t, c_t) = \mathrm{LLM}_{verify}(u, q, \mathcal{E}_t).$$

Here, $\hat{y}_t \in \{\text{TRUE}, \text{FALSE}, \text{ABSTAIN}\}$. Following Xie et al. (2025), we use a confidence threshold τ and accept a TRUE or FALSE label only when $c_t \geq \tau$. If $\hat{y}_t$ is ABSTAIN or $c_t < \tau$, the verifier treats the evidence as insufficient and performs another retrieval round, up to $T_{\max} = 5$. If no confident verdict is reached, a final prompt returns

$$\hat{y} = \mathrm{LLM}_{\text{final}}(u, q, \mathcal{E}_{T_{\max}}),$$

with the prompt biased toward ABSTAIN rather than an unsupported TRUE or FALSE. In the abstention-dropped setting, units with ABSTAIN labels or confidence below τ are left unscored. We also evaluate a calibrated-ensemble variant, where uncertain units fall back to the single-call baseline. Section 5.4 reports the resulting coverage-F_1^F trade-off. Full Prompts are provided in Appendix B.

5 Experiments and Results

We evaluate whether snippet-level verification improves factuality measurement across datasets, verifier models, and clinical structures. Unless otherwise stated, experiments use CLAIM-ONLY verification and report false-class F1, denoted F_1^F.

5.1 Experimental Protocol

We evaluate on the three datasets defined in Table 2, MEDSNIP-BENCH, HEALTHFC, and MEDHALLU. On the external datasets, we use the atomize-then-group pipeline to produce snippets before verification.

We evaluate six verifier models. The closed models are GPT-5.4 with high reasoning effort and GPT-4O with temperature zero. (Hurst et al., 2024) The open-weight models are Gemma-4-31B, GPT-OSS-20B, Llama-3.3-70B, and Llama-3.1-8B (Google, 2026; Agarwal et al., 2025; Grattafiori et al., 2024). All models use the same verification prompts. Atom and snippet decompositions are produced with GPT-5.4, so the comparison isolates the unit of verification.

For HEALTHFC and MEDHALLU, unit-level verdicts are aggregated with OR-FALSE, where an answer is predicted FALSE if any unit is predicted FALSE. For MEDSNIP-BENCH, gold labels are at snippet level, so no answer-level aggregation

Model	MEDSNIP-BENCH					HEALTHFC			MEDHALLU		
	atom	snippet	Δ	Mode 1	Mode 2	atom	snippet	Δ	atom	snippet	Δ
GPT-5.4-high	0.321	0.431	**+0.111**	0.429	0.419	0.630	0.670	**+0.040**	0.694	0.749	**+0.056**
GPT-4o	0.344	0.401	**+0.057**	0.301	0.317	0.632	0.698	**+0.066**	0.559	0.586	**+0.028**
gemma-4-31B-it	0.340	0.389	+0.049	0.350	0.378	0.606	0.699	**+0.093**	0.612	0.672	**+0.060**
gpt-oss-20b	0.344	0.401	**+0.057**	0.330	0.366	0.563	0.715	**+0.153**	0.623	0.645	**+0.022**
Llama-3.3-70B	0.316	0.331	+0.014	0.235	0.280	0.608	0.660	**+0.052**	0.507	0.515	+0.008
Llama-3.1-8B	0.238	0.278	+0.039	0.201	0.212	0.556	0.611	**+0.055**	0.471	0.497	**+0.026**

Table 3: F_1^F under CLAIM-ONLY verification across six verifier models and three datasets. Bold Δ marks significance under 95% BCa confidence intervals.

is required. Auto-snippets on MEDSNIP-BENCH use the drop-mixed label projection from human atom labels (Appendix C). Statistical significance is assessed with 95% BCa bootstrap confidence intervals using 10,000 resamples.

5.2 Snippet Verification Improves F_1^F

Table 3 compares atom- and snippet-level verification across the six verifier models and three datasets. On MEDSNIP, the main snippet column uses human-annotated snippets, while HEALTHFC and MEDHALLU use pipeline-generated snippets.

Snippet-level verification improves the point estimate in all 18 model-by-dataset cells, and the gap is significant under 95% BCa confidence intervals in 14 of 18. The non-significant cells are concentrated on MEDSNIP, where the sample is smaller, and on Llama-3.3-70B for MEDHALLU. The direction is stable across model strength. The weakest model still improves on every dataset, while GPT-5.4-high shows the largest MEDSNIP-BENCH gain.

The table also compares automatic snippets with human snippets on MEDSNIP-BENCH. For GPT-5.4-high, atomize-then-group snippets nearly match human snippets (0.429 vs. 0.431) and remain above atom-level verification (0.321). For weaker verifiers, automatic snippets are less stable, especially for Llama models. Snippet-direct narrows this degradation relative to atomize-then-group for all six models, suggesting that the best automatic segmentation strategy can depend on verifier strength.

These F_1^F gains also require fewer verifier calls. Snippet-level verification reduces calls by 22% on HEALTHFC, 26–32% on MEDSNIP-BENCH, and 47% on MEDHALLU.

Result Analysis: The cross-model results support the main granularity claim. Snippet-level verification not only reduces calls by grouping units, but also improves F_1^F across verifier families, including strong closed models and smaller open-weight models. This suggests that the benefit comes from preserving clinically meaningful local context, not from one verifier's behavior. At the same time, the automatic-snippet columns show that segmentation quality matters. The strongest verifier is less affected by pipeline-generated snippets, while weaker verifiers show larger drops from human-snippet performance. Thus, snippet-level verification is a granularity choice whose benefit depends on both the verification unit and the verifier's ability to reason over it. The advantage is not unconditional. Under FULL-CONTEXT verification with weaker models, atoms can match or beat snippets (Appendix C).

Pattern	n	Snip	Atom	Δ
A enumeration	1008	0.353	0.293	**+0.059**
B causal/cond.	225	0.387	0.304	**+0.083**
C concl.+prem.	321	0.442	0.391	+0.051
D standalone	617	0.384	0.303	**+0.081**
E topic-shift	39	0.143	0.211	−0.068
F distinct fact	314	0.642	0.586	+0.056
Overall	2524	0.427	0.359	**+0.068**

Table 4: Per-pattern F_1^F on MEDSNIP-BENCH using GPT-5.4 under CLAIM-ONLY verification. Bold Δ marks significance under 95% BCa confidence intervals.

5.3 Per-Pattern Analysis

The structural pattern annotations in MEDSNIP-BENCH let us examine where snippet-level verification helps. This analysis uses GPT-5.4 under CLAIM-ONLY verification, the setting where the snippet advantage is strongest. Table 4 reports per-pattern F_1^F by A–F pattern code.

Snippet-level verification is directionally better on all patterns except the small topic-shift category E. Among merge patterns, the largest gain is on Pattern B, causal-conditional chains, with $\Delta = +0.083$. This supports the hypothesis that

Metric	Baseline	Verifier, full	$\tau \geq 0.85$	$\tau \geq 0.90$
Dev F_1^F	0.518	0.518	0.562	**0.609**
Dev cov.	100%	96%	83%	63%
Test F_1^F	0.491	0.497	0.504	**0.552**
Test cov.	100%	97%	83%	61%

Table 5: Coverage-F_1^F trade-off on MEDSNIP-BENCH with GPT-5.4. Thresholded columns use abstention-dropped scoring, where abstained or low-confidence units are left unscored.

atomization is most harmful when verification depends on relations between clauses. Pattern E is the only category where atom-level verification is higher, but it contains only 39 snippets and the confidence interval crosses zero.

The single-atom patterns D and F require separate interpretation. Here, snippet and atom boundaries often coincide, so gains can reflect cleaner snippet text rather than a pure granularity effect. The merge-pattern gains on A, B, and C are the cleaner test of the granularity hypothesis. The per-pattern advantage is regime-dependent. Under FULL-CONTEXT or weaker verifiers, several gaps narrow or reverse, with Pattern B flipping under FULL-CONTEXT GPT-4O (Appendix C).

Result Analysis: The pattern-level results clarify why snippets help. The largest merge-pattern gain occurs on causal-conditional chains, where a clinical conclusion depends on linked premises, conditions, or mechanisms. These are the cases where atomization can separate the statement being checked from the information that makes it verifiable. Enumeration and conclusion-with-premises patterns also favor snippets, though with smaller gains. The positive deltas for standalone and distinct-fact patterns should be interpreted more cautiously, since those gains may reflect cleaner snippet wording rather than grouping itself. Overall, the pattern analysis supports the central mechanism that snippets help most when the clinical judgment is distributed across related clauses.

5.4 Retrieval-Augmented Verification Provides a Coverage Lever

Table 5 compares the retrieval-augmented verifier with the strong single-call baseline on MEDSNIP-BENCH dev and test. At near-full coverage, the retrieval verifier matches but does not improve over the baseline. Raising the confidence threshold trades coverage for higher F_1^F.

At $\tau = 0.90$, F_1^F reaches 0.609 on dev at 63% coverage and 0.552 on test at 61% coverage. The robust verifier finding is therefore a coverage-F_1^F trade-off, not a full-coverage improvement. Exploratory full-coverage ensembles and subset routers improved dev results but did not replicate on test, so we report them as negative results in Appendix D.

Results Analysis: Retrieval augmentation is useful, but for a different reason than the snippet result. It does not replace the single-call baseline at full coverage. Instead, it provides a confidence-controlled abstention mechanism. When the verifier is allowed to leave uncertain cases unscored, F_1^F rises consistently on both dev and test, but coverage falls. This makes the retrieval-augmented verifier useful for deployment settings where evidence and abstention are valuable, while keeping the main granularity finding independent of retrieval. The fact that the full-coverage variants did not replicate also argues against treating retrieval augmentation as the source of the paper's main improvement.

6 Conclusion and Future Work

Medical fact-checking depends not only on the verifier, but also on the unit being verified. We introduced MEDSNIP-BENCH, a human-annotated benchmark for snippet-level medical fact-checking, and MEDSNIP, an automatic pipeline for generating snippet-level verification units. Across MEDSNIP-BENCH, HEALTHFC, and MEDHALLU, snippet-level verification improves false-class F1 across verifier families while reducing verifier calls. The strongest and most interpretable gains occur when atomization separates causal, conditional, or premise-supported clinical reasoning into isolated claims. These results suggest that verification granularity should be treated as a core design choice in medical fact-checking pipelines, not a fixed preprocessing default.

In future work, we plan to explore adaptive granularity policies that choose between atomic, snippet-level, and document-level verification based on clinical reasoning structure and verifier uncertainty, rather than relying on a fixed decomposition strategy. We further plan to extend MEDSNIP beyond English and integrate richer clinical evidence sources, such as guidelines, electronic health record–style data, and longitudinal patient context, to better evaluate factuality in realistic clinical decision-support settings.

Limitations

Scope. Our experiments focus on English medical fact-checking datasets. Other languages and domains may require different granularity choices, especially when claims are more independent or when retrieval coverage differs substantially. We view snippet-level verification as a design choice for medically structured claims, not as a universal replacement for atom-level verification.

Model and aggregation dependence. The main experiments cover six verifier models, with GPT-5.4 used for decomposition across datasets. Absolute F_1^F scores depend on verifier strength, and the relative atom–snippet gap can vary with prompt context and aggregation rule. In particular, weak verifiers in FULL-CONTEXT mode can reduce or reverse the snippet advantage. We report aggregation sensitivity and verifier ablations in Appendix C.

Granularity and text quality. For some single-atom patterns, snippet and atom boundaries coincide. In these cases, gains may reflect cleaner snippet wording, such as resolved references and more self-contained text, rather than granularity alone. Our strongest evidence for the granularity claim comes from merge patterns such as enumeration, causal or conditional chains, and conclusion-with-premises structures.

Pipeline and retrieval design. External-dataset snippets are generated automatically with a single LLM-based decomposer. A different decomposer could change absolute performance, though the call-count reduction follows directly from producing fewer verification units. Similarly, our retrieval-augmented verifier is intentionally simple and uses general web and biomedical-literature retrieval rather than specialized clinical-evidence sources. Stronger retrieval may improve coverage and absolute F_1^F, but retrieval design is not the main focus of this work.

Development and test use. We use the held-out test split to confirm patterns observed during development. Two development-time variants, a calibrated ensemble and a subset-conditional router, did not replicate on test. We report these negative results in Appendix D.

Ethics and Broader Impact

This work studies automated factuality evaluation for medical text generation. It is intended to support auditing and research on medical LLMs, not to provide medical advice, diagnosis, treatment recommendations, or autonomous clinical decision support. The systems evaluated here should be used only with appropriate expert oversight.

A central motivation is harm reduction. Unsupported medical recommendations can mislead patients, clinicians, and downstream systems, and more reliable fact-checking may help identify such failures earlier. At the same time, automated verification is imperfect. A verifier may accept false claims, reject correct claims, or abstain on clinically important cases. Our results should therefore be interpreted as progress on evaluation methodology, not as evidence that automated medical fact-checking is solved.

MEDSNIP is derived from previously released medical QA resources and does not contain real patient records, protected health information, or identifiable clinical documentation. The clinical-vignette examples are synthetic educational cases rather than electronic health records. Nevertheless, the dataset may reflect biases in medical education materials, English-language evidence sources, and annotation practices.

The annotation task also involves judgment calls about snippet boundaries and contextual correctness. We provide guidelines and report inter-annotator agreement, but other annotator groups or healthcare settings may choose different decompositions. For this reason, we release the annotation guidelines together with the data and code to support inspection and reuse.

More efficient verification could also be misused to scale low-quality medical content or to create overconfident evaluation pipelines. We recommend treating snippet-level verification as an assistive auditing signal rather than a substitute for expert review.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, and 106 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025. Benchmarking large language models on answering and explaining challenging medical

questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599, Albuquerque, New Mexico. Association for Computational Linguistics.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. FacTool: Factuality detection in generative AI – A tool augmented framework for multi-task and multi-domain scenarios. *Preprint*, arXiv:2307.13528.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Google. 2026. Gemma 4 model card. `https://ai.google.dev/gemma/docs/core/model_card_4`. Accessed 2026-05-23.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jiayi He, Yangmin Huang, Qianyun Du, Xiangying Zhou, Zhiyang He, Jiaxue Hu, Xiaodong Tao, and Lixian Lai. 2025. Medfact: Benchmarking the fact-checking capabilities of large language models on chinese medical texts. *Preprint*, arXiv:2509.12440.

Heyuan Huang, Alexandra DeLucia, Vijay Murari Tiyyala, and Mark Dredze. 2025. Medscore: Generalizable factuality evaluation of free-form medical answers by domain-adapted claim decomposition and verification. *Preprint*, arXiv:2505.18452.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 399 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Hasan Iqbal, Yuxia Wang, Minghan Wang, Georgi Nenkov Georgiev, Jiahui Geng, Iryna Gurevych, and Preslav Nakov. 2024. OpenFactCheck: A unified framework for factuality evaluation of LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 219–229, Miami, Florida, USA. Association for Computational Linguistics.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Hyunjae Kim, Jiwoong Sohn, Aidan Gilson, Nicholas Cochran-Caggiano, Serina Applebaum, Heeju Jin, Seihee Park, Yujin Park, Jiyeong Park, Seoyoung Choi, Brittany Alexandra Herrera Contreras, Thomas Huang, Jaehoon Yun, Ethan F. Wei, Roy Jiang, Leah Colucci, Eric Lai, Amisha Dave, Tuo Guo, and 8 others. 2025. Rethinking retrieval-augmented generation for medicine: A large-scale, systematic expert evaluation and practical insights. *Preprint*, arXiv:2511.06738.

Yining Lu, Noah Ziems, Hy Dang, and Meng Jiang. 2025. Optimizing decomposition for optimal claim verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5095–5114, Vienna, Austria. Association for Computational Linguistics.

Itay Manes, Naama Ronn, David Cohen, Ran Ilan Ber, Zehavi Horowitz-Kugler, and Gabriel Stanovsky. 2024. K-QA: A real-world medical Q&A benchmark. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 277–294, Bangkok, Thailand. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

OpenAI. 2026. Introducing GPT-5.4. `https://openai.com/index/gpt-5-4-thinking-system-card/`. Accessed 2026-05-23.

Shrey Pandit, Jiawei Xu, Junyuan Hong, Zhangyang Wang, Tianlong Chen, Kaidi Xu, and Ying Ding. 2025. MedHallu: A comprehensive benchmark for detecting medical hallucinations in large language models. In *Proceedings of the 2025 Conference on*

Empirical Methods in Natural Language Processing, pages 2858–2873, Suzhou, China. Association for Computational Linguistics.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, and 13 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R. Pfohl, Heather Cole-Lewis, Darlene Neal, Qazi Mamunur Rashid, Mike Schaekermann, Amy Wang, Dev Dash, Jonathan H. Chen, Nigam H. Shah, Sami Lachgar, Philip Andrew Mansfield, and 16 others. 2025. Toward expert-level medical question answering with large language models. *Nature Medicine*, 31(3):943–950.

Juraj Vladika, Phillip Schneider, and Florian Matthes. 2024. HealthFC: Verifying health claims with evidence-based medical fact-checking. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8095–8107, Torino, Italia. ELRA and ICCL.

Miriam Wanner, Seth Ebner, Zhengping Jiang, Mark Dredze, and Benjamin Van Durme. 2024. A closer look at claim decomposition. In *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, pages 153–175, Mexico City, Mexico. Association for Computational Linguistics.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. 2024. Long-form factuality in large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 80756–80827. Curran Associates, Inc.

Zhuohan Xie, Rui Xing, Yuxia Wang, Jiahui Geng, Hasan Iqbal, Dhruv Sahnan, Iryna Gurevych, and Preslav Nakov. 2025. FIRE: Fact-checking with iterative retrieval and verification. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2901–2914, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhihong Zhu, Yunyan Zhang, Xianwei Zhuang, Fan Zhang, Zhongwei Wan, Yuyan Chen, Qingqing Long, Yefeng Zheng, and Xian Wu. 2025. Can we trust AI doctors? a survey of medical hallucination in large language and large vision-language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6748–6769, Vienna, Austria. Association for Computational Linguistics.

A Data Construction Details

This appendix documents the construction of the MEDSNIP benchmark: how we partitioned the parent corpus, how we split into train, dev, and test, what inter-annotator agreement looks like, and how faithfully the MEDSNIP pipeline reproduces the human snippet structure.

A.1 Subset Selection

The Kim et al. (2025) corpus blends short consumer-health questions from K-QA (Manes et al., 2024) with longer USMLE-style clinical-vignette questions from MEDBULLETS (Chen et al., 2025). Query length is bimodal, consumer-health queries range from 2 to 31 words and clinical-vignette questions range from 112 to 254 words, with no queries in the 32–111-word gap (Figure 3a). We use a word-length threshold at 80 to split the 276 entries into 140 consumer-health and 136 clinical-vignette entries. The natural gap means the partition is unambiguous, any threshold between 32 and 111 yields the same split.

The two subsets differ qualitatively in the verification demand they place on a fact-checker, not just in length. Consumer-health queries are short, often single-sentence questions about a drug, dose, interaction, or symptom, and the answers tend to be lay-readable explanations grounded in standard-of-care knowledge. Clinical-vignette queries supply a structured patient history (demographics, presenting complaint, history, medications, vitals, and findings) and ask for a diagnostic conclusion, next test, or treatment, so the answer chains together premises whose correctness depends on the patient's specific frame rather than on general medical facts alone.

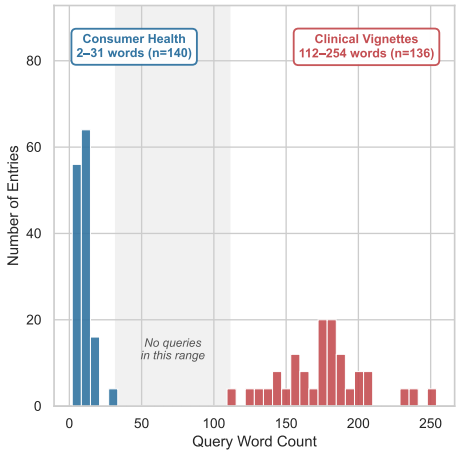
Representative queries. A typical consumer-health query is a single short sentence:

> *"Can I take Nyquil and Benadryl at the same time?"*

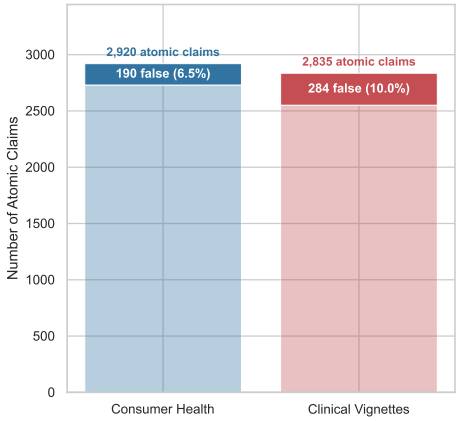
A typical clinical-vignette query is a paragraph-length scenario followed by a question:

> *"A 1-year-old girl is brought to a neurologist due to increasing seizure frequency over the past 2 months. She recently underwent a neurology evaluation which revealed hypsarrhythmia on EEG with multifocal spikes... Her medications consist of lamotrigine and valproic acid... What is the most appropriate next step in management?"*

The two query styles drive snippet patterns: consumer queries skew toward Pattern A (enumeration

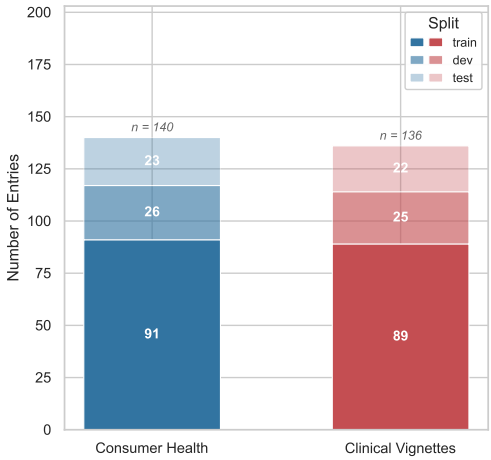


(a) Query word-length distribution.

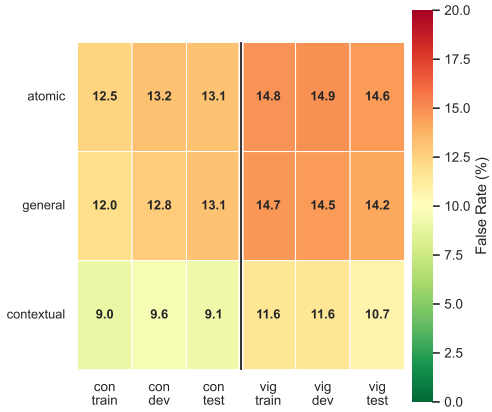


(b) Atomic-claim composition by subset.

Figure 3: Subset selection from the parent corpus.



(a) Per-subset entry counts across splits.



(b) Snippet-level false-rate per (subset × split).

Figure 4: Train / dev / test split.

of features or factors), while vignettes skew toward Patterns B and C (causal–conditional chains and conclusion-with-premises). The pattern distribution in Table 1 reflects this asymmetry.

Vignette responses carry a higher atom-level false-rate (10.0%) than consumer responses (6.5%), consistent with the harder clinical-reasoning content (Figure 3b). All downstream subset-conditional analyses preserve this distinction.

A.2 Train / Dev / Test Split

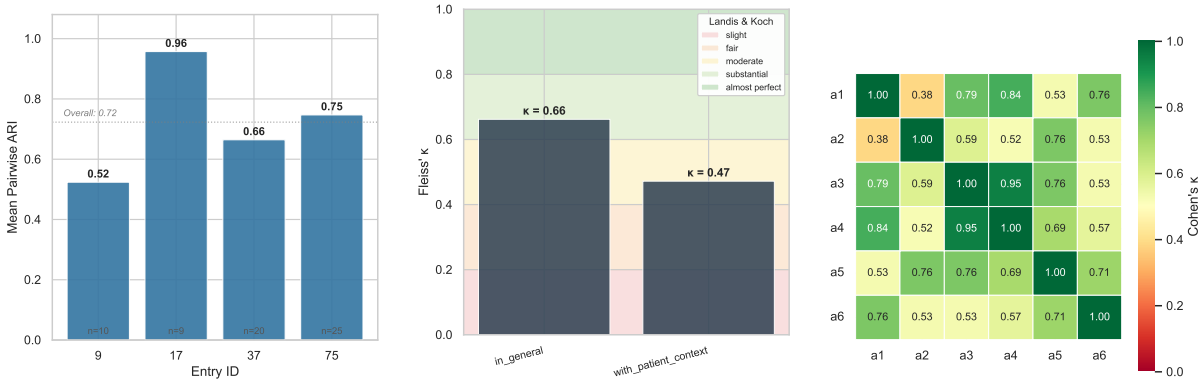
We stratify the 276 entries on the cross of source subset (consumer vs. vignette) and a coarse in-general false-rate bucket, then sweep random seeds to minimize the worst per-(subset × split) snippet-level false-rate spread across the three label dimensions (atomic, in-general, in-patient-context). Seed 202 attains a worst-case spread of 1.1 percentage points, which we adopt. Headline entry and snippet counts per split are reported in Table 1.

The resulting partition preserves the original consumer-to-vignette ratio across splits (Figure 4a). Snippet-level false-rates are tightly matched across splits within each subset on all three label dimensions (Figure 4b).

A.3 Shared Context

Each entry includes a *shared context* block, a structured key–value summary of the query and response framing that applies to all snippets. It anchors decontextualized snippet text in the clinical or topical scenario, supporting consistent verification against the relevant patient or query.

The schema is subset-dependent. Clinical-vignette entries use keys such as `age`, `sex`, `chief_complaint`, `medical_history`, `medications`, `vitals`, `treatment_history`, and `correct_answer`. Consumer-health entries use lighter keys such as `topic` and `drug_a/drug_b`. The schema is open, annotators add keys when an entry introduces information that downstream snippets refer to.



(a) Grouping agreement (ARI per entry). (b) Fleiss' κ per labeled field. (c) Pairwise Cohen's κ for in-general.

Figure 5: Inter-annotator agreement on the 64-snippet IAA batch annotated independently by all six annotators.

The MEDSNIP pipeline preserves this convention. Mode 1 and Mode 2 both emit a shared-context block alongside the snippet list (Section 4.1), so automatic and human-annotated entries are interchangeable downstream. Verification prompts may reference the shared context to recover entities that are absent from the snippet text, which matters most for the with-patient-context label where the relevant frame is not in the snippet itself.

A.4 Inter-Annotator Agreement

The pattern label distribution and inter-annotator reliability statistics for MEDSNIP were computed on a 64-snippet IAA batch annotated independently by all six annotators. Figure 5 visualizes the three views of agreement that the rest of this subsection discusses: chunking agreement per entry, Fleiss' κ on each labeled field, and the pairwise Cohen's κ matrix on the headline in-general label.

Binary in-general label. Fleiss' $\kappa = 0.662$ over the six-annotator multi-set on the 64-snippet IAA batch (Figure 5b). Per-pair Cohen's κ ranged from 0.385 to 0.947 across the 15 annotator pairs (mean 0.663, see Figure 5c). Exact-agreement-among-all-6 was achieved on 52 of 64 items (81.3%).

Snippet chunking. Adjusted Rand index (ARI) on the snippet-partition decision, computed by pairing the six annotators on the same 64 items, gave a mean ARI of 0.723 (range $[0.524, 0.957]$ across pairs; per-entry distribution in Figure 5a). The chunking decision is more constrained than the binary labeling and shows substantial agreement.

Pattern label. The pattern annotation was integrated into the same annotation pass under the written guidelines (Appendix E). On a held-aside subset where multiple annotators tagged the same snippet's pattern, per-pattern agreement was $\kappa \approx 0.59$ (substantial), with the largest residual confusions between A (enumeration) and C (conclusion + premises) on snippets that resemble both. The full taxonomy and worked anchor cases were recirculated as a calibration step before the production pass.

Mode	sent F1	sent P	sent R	embed cos
Mode 1	0.756	0.723	0.898	0.824
Mode 2	0.734	0.744	0.793	0.794

Table 6: Auto-snippet pipeline chunking fidelity against the human gold structure on all 276 entries.

A.5 Auto-Snippet Pipeline Quality

The MEDSNIP pipeline reproduces snippet structure automatically. We evaluate how faithfully its output matches the human-annotated MEDSNIP benchmark across all 276 entries. Two metrics are relevant: *chunking fidelity* (does the pipeline draw snippet boundaries where humans drew them?) and *downstream label robustness* (do verification scores depend on the choice of label-projection policy for auto-snippets?).

Chunking fidelity. Sentence-set F1 against the human gold chunking is 0.756 for Mode 1 (atom-to-snippet) and 0.734 for Mode 2 (snippet-direct). Embedding cosine similarity gives 0.824 and 0.794. Mode 1 preserves more of the human structure on average, in line with its two-pass design that respects atom-level boundaries (Table 6).

Snippet count. The pipeline under-segments relative to humans. Auto snippets total 1,578 in Mode 1 and 2,132 in Mode 2, compared with 2,524 human gold snippets, giving ratios of 0.62 and 0.84. Consumer entries are reproduced well, with sentence F1 above 0.85 in both modes, while vignettes are harder at ≈ 0.65 in both modes. This is consistent with the harder clinical-reasoning content noted in the subset-selection analysis above.

Label projection sensitivity. Auto snippets inherit a label projected from the human atom-level labels of the source atoms. We compare three policies: **AND** (all covered atoms must be true, with mixed-label sentences treated as false), **majority** (majority vote on covered atoms, ties resolved as true), and **drop-mixed** (drop any auto-snippet whose source atoms straddle sentence regions with conflicting human labels). Drop-mixed policy is primary throughout the main body. The AND and majority policies are robustness checks and give qualitatively the same picture, with positive snippet-vs-atom direction across all six models on both modes; the projection bias is small relative to the model-level differences. Coverage for the drop-mixed policy is 1,382 kept snippets in Mode 1 and 1,753 in Mode 2; the dropped fractions reflect mixed-label coverage and not pipeline failures.

B Decomposition and Verification Prompts

The prompts are built from reusable guideline, schema, formatting, granularity, and task-instruction components. We describe each component once and provide a composition map linking production prompts to their components.

B.1 Single-Call Baseline Verifier

The baseline verifier (Section 4.2) takes one unit and returns a binary verdict in a single LLM call. Atom and snippet baselines share these prompts; the unit text field is `claim` for atoms and `snippet_text` for snippets. Figures 6 and 7 show both prompts verbatim.

B.2 MedSNIP Pipeline: Prompt Composition

The pipeline (Section 4.1) uses three system prompts: Mode 1 extract, Mode 1 cluster, and Mode 2 direct. Each prompt is assembled from the reusable blocks in Table 7. Consumer and vignette variants use the same prompt skeleton, with subset-specific schemas, text rules, and granularity rules. The five reusable blocks are shown in Figures 8–12.

Component	M1 Extract	M1 Cluster	M2
Merge patterns	–	✓	✓
Atom extraction	✓	–	–
Granularity rules	✓	–	–
Context schema	✓	✓	✓
Snippet text rules	–	✓	✓
Output JSON	extract	cluster	direct

Table 7: Reusable component composition

CLAIM-ONLY baseline

System Prompt:
You are a cautious medical fact-checking assistant. Given a claim, decide if the claim is true. Respond with exactly one word: `"true"` if the claim is fully supported, otherwise `"false"`. No other words.

User Message:

```
Claim:
{snippet_text}
```

Figure 6: Baseline verifier in CLAIM-ONLY mode.

FULL-CONTEXT baseline

System Prompt:
You are a cautious medical fact-checking assistant. You will be given a user question, a full answer text, and a single extracted claim (decomposed from the full answer text). Use the full answer text ONLY to decontextualize the claim. Then decide whether the interpreted claim is medically correct in the real world. Do NOT treat the full answer text as evidence the claim is true; it is context for interpretation. Respond with exactly one word: `"true"` or `"false"`. No other words.

User Message:

```
Question:
{query}

Full answer text (interpretation only):
{full_text}

Extracted claim:
{snippet_text}
```

Figure 7: Baseline verifier in FULL-CONTEXT mode. The full answer is for disambiguation only.

Block: Merge Patterns A–F

You mirror a medical-text annotator. Group an LLM-generated medical response into **snippets**, coherent, self-contained, verifiable units. Guiding question for every claim: *If this stood alone, could a reader verify it medically?* If no → merge.

Three patterns that trigger MERGE

`<Pattern A-C from Section 3>`

Three patterns that justify KEEPING ATOMIC

`<Pattern D-F from Section 3>`

Default. When torn, keep atomic.

Coverage: every unit appears in exactly one snippet.

Figure 8: Shared annotation guidelines, used by Mode 1 cluster and Mode 2.

Block: Shared-context schemas

Consumer Keys: topic, drug_a, drug_b, condition, symptom. Omit absent keys; values are short and concrete.

Vignette Keys: age, sex, chief_complaint, medical_history, medications, vitals, exam_findings, treatment_history, correct_answer. Omit absent keys.

Figure 9: Per-subset `shared_context` dict shapes. Plugged into every pipeline prompt.

Block: Snippet text rules

Consumer. Each snippet's `output` is a self-contained verifiable statement. A reader should judge correctness without the query. Do NOT introduce facts absent from the source.

Vignette. Each snippet's `output` must be self-contained. **Inline relevant patient demographics** from `shared_context` so a reviewer can judge it standalone (e.g., "this patient" → "this 55-year-old male with right arm weakness"). Do NOT introduce facts absent from the source.

Figure 10: Per-subset snippet wording rules. Used by Mode 1 cluster and Mode 2.

Task Heads and Output Schemas

Each prompt ends with a mode-specific task and JSON schema (Figures 13–15).

B.3 Retrieval-Augmented Verifier

The retrieval verifier (Section 4.3) runs an iterative loop. The system prompt is a single concatenation of the blocks in Figures 16–19. When the iteration cap is reached without a final verdict, the force-final fallback in Figure 19 is used instead.

Block: Granularity rules

Consumer (fine). "X can cause drowsiness, dizziness, and confusion" → three atoms, one per side effect. "Drug X is an antihistamine containing compound Z" → one atom. Single-property sentences → one atom.

Vignette (coarse). One atom per sentence by default. Causal chains and multi-property sentences stay as one atom. The final-answer sentence is its own atom. Each differential-option sentence becomes one atom.

Figure 11: Per-subset atom granularity rules. Used by Mode 1 step 1 (extract).

Block: Atom-extraction

You are an atomic-claim extractor. Given a user query and an LLM response pre-segmented into numbered sentences, produce a `shared_context` dict plus a list of atomic claims that follow the sentence structure.

Atom definition. Each atom states ONE medically-meaningful proposition, light-normalized from its sentence(s). Resolve pronouns. **Preserve the response's stance** — do NOT correct claims that look wrong.

Sentence-atom mapping. One sentence → one atom by default; one sentence → multiple atoms only for clear comma-separated enumerations; two consecutive sentences → one atom only for indivisible claims.

Coverage. Every sentence index must appear in at least one atom's `source_sentences`.

Figure 12: Atom-definition and sentence-mapping rules. Used by Mode 1 step 1 (extract).

Task head: Mode 1 step 1 (extract)

Task. Use the granularity rule for the subset; emit `shared_context` per the schema; cover every sentence index.

Output (JSON only):

```
{
  <JSON schema>
}
```

Figure 13: Mode 1 step 1 task and output. Combines with the extraction-common block, granularity, and the per-subset context schema.

C Experimental Details

This appendix documents the experimental setup, statistical methodology, full per-cell numbers, and sensitivity analyses behind the main-body results.

Task head: Mode 1 step 2 (cluster)

Task. Given the query, the pre-extracted `shared_context`, and a numbered list of `atomic_claims`, group atoms into snippets per Patterns A–F. Every atom index appears in exactly one snippet. Write each snippet's `output` per the text rules; tag the dominant pattern (A–F) and a short `notes` rationale.

Output (JSON only):

```
{ <JSON schema> }
```

Figure 14: Mode 1 step 2 task and output. Combines with the merge-patterns block, the per-subset context schema, and snippet text rules.

Task head: Mode 2 (snippet-direct)

Task. Given the query and numbered sentences, in one pass: (1) extract `shared_context`; (2) identify atomic claims and group them into snippets per Patterns A–F (every sentence index appears in at least one snippet); (3) write `output` per the text rules; (4) tag the dominant pattern.

Sentence-boundary default. The default is one sentence → one snippet. Merge only when an A/B/C cue is explicit across sentences; if not, split. Topical similarity alone is NOT enough. A common failure mode is collapsing the whole response into one snippet — do not do that. For N-sentence responses, the output usually has $0.6N$ to N snippets.

Output (JSON only):

```
{ <JSON schema> }
```

Figure 15: Mode 2 task and output. Combines with the merge-patterns block, the per-subset context schema, and snippet text rules.

Verifier: Answer, Abstain or Search

You are a rigorous medical fact-checker. You receive a snippet, the `shared_context`, and any evidence gathered so far. Your job is to **catch wrong claims** — false-negative detection is the critical metric.

Confidence: 0.90–1.00 direct specific evidence; 0.70–0.89 strong with minor uncertainty; 0.50–0.69 leaning but not confident; <0.50 search more. Be calibrated, not optimistic.

Output (JSON only):

```
{
  <JSON schema for Final Answer,
  Abstain and Search>
}
```

Figure 16: Verifier framing and the per-iteration JSON options with calibrated confidence bands.

Verifier: Must-Search, Refutation

Must-search triggers. Do not short-circuit on parametric knowledge when the snippet has specific numbers (doses, ranges, percentages, durations, thresholds), equivalence claims between drugs/conditions/mechanisms, vignette-style clinical reasoning, recent guidelines, or any uncertainty on a named drug/dose/condition.

Refutation-biased phrasing. Look for evidence the snippet is *wrong*, not for confirmation.

Source. `web` for consumer info, brand-name OTC, recent guidelines, recency-sensitive content. `pubmed` for primary clinical literature, RCT efficacy, mechanism, rare conditions; use keyword phrases, not full questions.

Figure 17: Verifier search behavior: when to issue a search, how to phrase it, and which retriever to call.

Verifier: General Truth Framing

General-truth framing. Judge the snippet as a *general* medical statement, NOT vignette applicability.

Compound-claim rule. A multi-sub-claim snippet is true only if every *material* sub-claim is true. Material errors include the wrong drug class, effect direction, mechanism, or target population. Ignore acceptable rounding, rare-exception generalizations, hedging, minor phrasing or mechanism imprecision, overlapping-category framing, and comparative edge cases when the core assertion holds.

Mixed evidence. If evidence neither clearly supports nor refutes a specific factual claim with named entities, numbers, or mechanism → **false**. Generic broadly-true statements with no specific content → **true** if nothing specific is wrong.

Figure 18: Verifier truth-judgement rules: general-medical truth, compound-claim materiality, and the mixed-evidence default.

Verifier: Force-final Fallback

You are a cautious medical fact-checker making a final decision; no more searches are allowed.

Prefer abstain over guessed False. If the evidence does not directly refute a material claim, return **abstain**. If the main assertion is defensible and only peripheral sub-claims are doubtful, return **true** under the compound-claim rule.

Only emit `final_answer: false` with concrete refuting evidence, such as a wrong entity, out-of-range number, inverted mechanism, or misassigned category.

Figure 19: Force-final fallback prompt used at the iteration cap. Biased toward abstention on residual uncertainty.

C.1 Compute and API

Closed-source models: GPT-5.4 (with `reasoning_effort=high`) and GPT-4O are accessed through the OpenAI API. GPT-5.4 is the canonical decomposer across atom and snippet conditions to keep the comparison free of atomizer-quality confounds. Per-experiment dollar costs for both closed-source models are in Table 8.

Open-weight models: GEMMA-4-31B-IT, GPT-OSS-20B, LLAMA-3.3-70B-INSTRUCT, and LLAMA-3.1-8B-INSTRUCT are accessed through the HF Inference router with the OpenAI-compatible chat-completions endpoint, so the same prompt and parsing pipeline applies across all six models. End-to-end dollar costs per cell are reported in Table 8.

Retrieval backends: The retrieval-augmented verifier uses a general web retriever ($0.001 per query) and a public biomedical-literature API (free, abstracts only). Both return $k = 10$ snippets per query. Each verified item issues between one and five retrieval rounds depending on confidence-gated stopping, for an average of ≈ 2.3 retrievals per item.

Random seeds: All bootstrap analyses use a single fixed seed (`seed=42`). The train/dev/test partition was selected as the random-seed sweep winner (seed 202). Stochastic LLM calls use the API default sampling temperature.

C.2 BCa Bootstrap CI Methodology

All confidence intervals for F_1^F differences are 95% bias-corrected and accelerated (BCa) bootstrap intervals. We use $B = 10{,}000$ bootstrap replicates. Resampling is paired at the entry level: each replicate samples source entries with replacement and recomputes both atom-level and snippet-level F_1^F on the same sampled entries. Thus, the reported Δ is a paired difference between two statistics, not an unpaired effect size.

A cell is declared significant when the 95% BCa interval for Δ excludes zero. The "14 of 18 cells significant" result in Section 5.2 counts cells that satisfy this criterion across the three-dataset, six-model grid.

We use BCa intervals because the bootstrap distribution of Δ can be skewed, especially when one of the two F1 values is near zero in small pattern cells or weak-verifier settings. BCa intervals correct for both bias and skewness in the bootstrap distribution, making them more appropriate than simple percentile intervals for these comparisons.

C.3 Full Per-Cell Results

Table 8 provides the per-split MEDSNIP results underlying the aggregate numbers in Table 3. It reports atom and snippet verifier-call counts, false-class F1, and dollar cost for each evaluated split. Figure 20 visualizes the corresponding cost–F_1^F trade-off, showing how snippet-level verification shifts the baselines toward fewer verifier calls

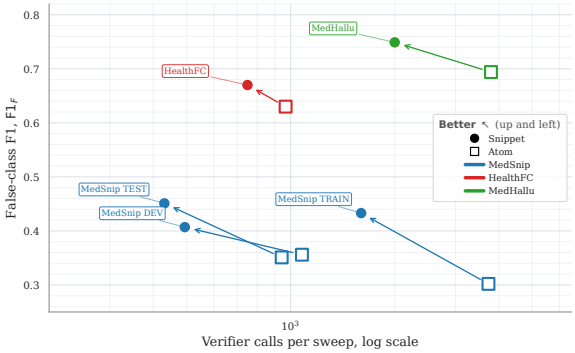


Figure 20: Verifier-call count versus F_1^F for snippet- and atom-level baselines across the three datasets. Snippet-level verification moves points leftward, indicating fewer verifier calls, while preserving or improving F_1^F.

C.4 Aggregation Rule Sensitivity

For datasets with answer-level labels, per-unit verdicts must be aggregated into a single answer-level prediction. We audit four aggregation rules: OR-FALSE (predict FALSE if any unit is false, following the FACTSCORE default), AND-FALSE (predict FALSE only if all units are false), MAJORITY (majority vote), and THRESHOLD-k with $k = 2$ and $k = 3$ (predict FALSE if at least k units are false). For each of the 14 dataset–granularity–mode–model settings, we compute the snippet–atom F_1^F gap and its 95% paired bootstrap confidence interval under each rule.

Findings: Aggregation affects the observed snippet–atom gap. Under THRESHOLD-k with $k \geq 2$, snippet-level verification exceeds atom-level verification in 12 of 14 settings with $P(\Delta > 0) \geq 0.95$. The two exceptions are both on MEDSNIP with GPT-4O in FULL-CONTEXT mode, where the verifier produces near-degenerate predictions. Under OR-FALSE, snippet-level verification wins in 9 of 14 settings. Under AND-FALSE, the sign reverses on MEDHALLU (atom > snippet by

Model	Split	Atoms	Snips.	Atom F_1^F	Snippet F_1^F	Δ	Atom Cost	Snippet Cost
MEDSNIP CLAIM-ONLY *verification*								
GPT-5.4-high	train	3735	1599	0.302	0.433	+0.131	$10.27	$7.13
	dev	1078	494	0.356	0.407	+0.051	$2.96	$2.20
	test	942	431	0.351	0.451	+0.100	$2.59	$1.92
GPT-4O	train	3735	1599	0.328	0.416	+0.088	$1.70	$0.80
	dev	1078	494	0.361	0.395	+0.034	$0.49	$0.24
	test	942	431	0.382	0.354	−0.028	$0.43	$0.22
gemma-4-31B-it	train	3735	1599	0.324	0.394	+0.070	$0.05	$0.02
	dev	1078	494	0.351	0.395	+0.045	$0.01	< $0.01
	test	942	431	0.383	0.358	−0.025	$0.01	< $0.01
gpt-oss-20b	train	3735	1599	0.323	0.400	+0.077	$0.10	$0.06
	dev	1078	494	0.355	0.398	+0.043	$0.03	$0.02
	test	942	431	0.406	0.408	+0.002	$0.02	$0.02
Llama-3.3-70B	train	3735	1599	0.299	0.322	+0.023	$0.05	$0.03
	dev	1078	494	0.326	0.369	+0.043	$0.02	< $0.01
	test	942	431	0.368	0.314	−0.054	$0.01	< $0.01
Llama-3.1-8B	train	3735	1599	0.216	0.287	+0.071	< $0.01	< $0.01
	dev	1078	494	0.239	0.269	+0.029	< $0.01	< $0.01
	test	942	431	0.316	0.255	−0.061	< $0.01	< $0.01
MEDSNIP FULL-CONTEXT *verification*								
GPT-5.4-high	train	3735	1599	0.475	0.475	−0.000	$17.67	$11.27
	dev	1078	494	0.485	0.518	+0.034	$5.10	$3.48
	test	942	431	0.491	0.491	−0.000	$4.46	$3.04
GPT-4O	train	3735	1599	0.338	0.301	−0.037	$7.12	$3.25
	dev	1078	494	0.321	0.227	−0.094	$1.97	$0.96
	test	942	431	0.222	0.135	−0.087	$1.79	$0.86

Table 8: Per-split MEDSNIP results. Atom and snippet columns report verifier-call counts. F_1^F columns report false-class F1 for atom- and snippet-level verification. Cost columns report dollar cost for one full sweep of the cell.

0.05, $P = 0.036$), because this rule requires every unit to be FALSE and atomization produces more units.

Implication: Aggregation is not a neutral implementation detail. Changing the rule can change the sign of a granularity comparison, so decomposition-based fact-checking results should report the aggregation rule explicitly. The cost reduction reported in the main paper is unaffected by this choice, since it is determined by the number of verification units.

D Retrieval-Augmented Verifier Development Checks

The main paper reports the final retrieval-augmented verifier and its coverage–F_1^F trade-off. During development, we also tested two variants intended to recover full coverage while preserving the benefits of confidence gating. The first used the retrieval verifier when its confidence exceeded the threshold and otherwise fell back to the single-call baseline. The second routed examples by source subset, using the retrieval verifier for one subset and the single-call baseline for the other. Both variants improved development performance but did not provide a stable improvement on the held-out test split. We therefore report the simpler confidence-thresholded verifier in the main paper and treat full-coverage routing or ensembling as future work rather than part of the core result.

E Annotation Guidelines (verbatim)

The text below is the guideline document distributed to annotators, reproduced with only the operational/identifying material removed (annotation-tool URLs, per-annotator batch assignments). The pattern taxonomy and the worked examples used as

inter-annotator anchors are reproduced unchanged.

E.1 Background

We have 5,755 atomic claims from 276 medical QA entries. The annotator's job is to **group these atoms into snippets** and **label each snippet** with dual labels. Each entry shows:

- **Query**: the medical question.
- **Full Response**: the LLM-generated answer.
- **Atomic Claims**: individual statements extracted from the response, each with an expert label (true / false).

Why we merge at all (the motivation). Atomic-claim extraction is often **too fine-grained**: a single atom like *"Option B is the correct answer"* or *"The factors include the individual's age"* is **unverifiable in isolation**, because it needs the surrounding reasoning to make sense. Medical responses, especially clinical vignettes, chain claims together with *because / since / therefore / requires*. Splitting those chains destroys the verifiable unit.

We merge atoms whose logical dependency would be lost if each stood alone. Every rule below, and every judgment call during consensus, traces back to one question:

> *"If I read only this atom (or only this snippet), do I have enough to check whether it's medically correct?"*

If **yes** → atomic is fine. If **no** → merge with the atoms that carry the missing context.

E.2 Task

For each entry, the annotator:

1. Defines **shared context**: key-value pairs describing the query/response context (patient demographics, medications, topic).
2. Selects atoms that belong together.
3. Creates a snippet by grouping them.
4. Writes the merged snippet text (an LLM draft is offered; the annotator edits as needed).
5. Sets two labels for each snippet:
   - **With context**: correct *for this specific patient/query*?
   - **In general**: correct as a *general medical statement*?
6. Tags the snippet's structural pattern (A–F).
7. Marks the snippet **ambiguous** if correctness cannot be determined.
8. Adds notes, especially when the two labels differ.

E.3 Shared Context

Shared context is defined **per entry** and applies to all snippets. For clinical vignettes, typical keys include *age*, *sex*, *chief_complaint*, *medical_history*, *medications*, *vitals*, *treatment_history*, *correct_answer*. For consumer-health queries, typical keys include *topic*, *drug_a*, *drug_b*.

E.4 Grouping Rules (Merge vs Keep Atomic)

Guiding question: *If this atom stood alone, could I still verify it?* If yes → keep atomic. If no → merge.

E.4.1 Three patterns that trigger MERGE

Pattern A: Enumeration of properties of one subject. The response lists multiple features, factors, or properties of the same thing. The enumeration is the verifiable unit.

Example A.1 — Consumer query on safe Advil dose:

- *atom 1:* "The safe amount of Advil (ibuprofen) to take at one time depends on several factors."
- *atom 2:* "The factors include the individual's age."
- *atom 3:* "The factors include the individual's weight."
- *atom 4:* "The factors include the individual's overall health."

Decision: MERGE. Atoms 2–4 begin with "The factors include…" and are meaningless without atom 1.

Example A.2 — Vignette on a 72-year-old man with suspected leukemia:

- *atom 10:* "Chronic Lymphocytic Leukemia (CLL) typically presents with a high leukocyte count."
- *atom 11:* "CLL typically presents with anemia."
- *atom 12:* "CLL typically presents with thrombocytopenia."

Decision: MERGE. Three clinical features that together form the "CLL picture".

Pattern B: Causal or conditional chain. Atoms connected by *because / since / as / due to / requires / therefore*, or where one atom is the premise for another. Merge so the logical link survives the decomposition.

Example B.1 — Vignette on a 23-year-old with treatment-resistant schizophrenia:

- *atom 7:* "Clozapine is generally reserved for patients who have failed to respond to at least two previous adequate trials of antipsychotic medications."
- *atom 8:* "Patients must have failed at least one conventional antipsychotic."
- *atom 9:* "Patients must have failed at least one second-generation antipsychotic."

Decision: MERGE. Atom 7 states the condition; atoms 8 and 9 specify the two required failures.

Example B.2 — Consumer query on Hepatitis A IgM:

- *atom 7:* "A non-reactive result does not necessarily rule out a past infection."
- *atom 8:* "IgM antibodies may not be detectable after a certain period."

Decision: MERGE. Atom 8 is the *reason* for atom 7 (non-reactive doesn't rule out past infection *because* IgM fades).

Pattern C: Conclusion + supporting premises. A recommendation, diagnosis, or final answer that is only meaningful when read with the reasoning that produced it.

Example C.1 — Same schizophrenia vignette, end of response:

- *atom 19:* "The most appropriate next step in management is to initiate clozapine."
- *atom 20:* "Option B is the correct answer."

Decision: MERGE. Atom 20 is unverifiable alone; *what* is Option B? Atom 19 provides the referent.

Example C.2 — Same vignette, earlier in response:

- *atom 1:* "The patient's aggressive behavior is likely secondary to schizophrenia."
- *atom 2:* "The patient's aggressive behavior could be due to a psychotic disorder."
- *atom 3:* "The most appropriate next step in management is to consider an alternative antipsychotic medication."

Decision: MERGE. Atoms 1–2 give the differential; atom 3 is the conclusion drawn from it.

E.4.2 Three patterns that justify KEEPING ATOMIC

Pattern D: Complete standalone fact, definition, or lab interpretation. The atom reads as a self-contained medical statement; no surrounding text needed.

Example D.1 — Hepatitis A IgM definition:

- *atom 1:* "A non-reactive result for the Hepatitis A IgM (Immunoglobulin M) antibody test indicates that the individual does not have a recent or current Hepatitis A infection."

Decision: ATOMIC. A complete definition; nothing else needed.

Example D.2 — Lab value interpretation:

- *atom 1:* "A platelet count of 119,000/mm^3 is within normal limits."

Decision: ATOMIC. A single lab-value claim verifiable in isolation. It happens to be **false** (the normal range is 150,000–400,000), but that is a labeling question, not a grouping one.

Pattern E: Topic genuinely shifts. The response moves from one subject to an unrelated one (e.g., drug composition → overdose risk, or one differential diagnosis → a different one). Split at the topic boundary.

Pattern F: Same topic, but each atom carries a different checkable fact. Two claims about the same drug can still be separate if they convey distinct information. A standalone fact does not need to be dragged into a nearby enumeration just because it sits next to one. The "isolated [false] atom" idiom (used to keep a false atom apart so it does not contaminate adjacent true claims) also falls under Pattern F.

E.4.3 Default: when in doubt, keep atomic

Default to **atomic**. It is cheaper and easier to merge during consensus review than to split a bad merge. If Pattern A, B, or C clearly fires, merge without hesitation.

E.4.4 Pattern Tagging

After the merge / keep-atomic decision, the annotator records the **dominant pattern code** (A–F) on the snippet. The code captures the structural reason for the chunking decision and enables per-pattern downstream analyses.

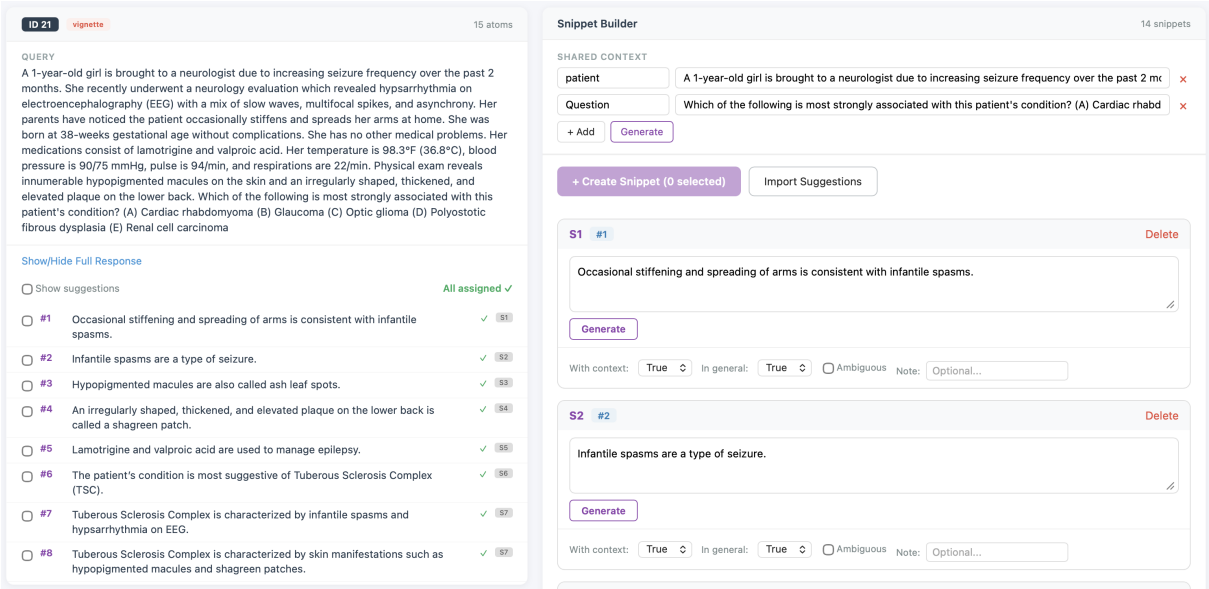


Figure 21: Annotation dashboard. Left panel: atomic claims with expert labels, query, and full LLM response (collapsed). Right panel: shared context plus per-snippet cards with atom selection, merged text, dual labels, ambiguity flag, notes, and pattern code (A–F). All annotator-identifying elements have been removed for review.

Selection rules.

- **Multi-atom snippets** (the result of a MERGE decision) receive one of **A**, **B**, or **C**.
- **Single-atom snippets** (the result of a KEEP-ATOMIC decision) receive one of **D**, **E**, or **F**.

Hybrid snippets. A snippet may carry an **optional secondary code** when two patterns clearly apply. For example, a snippet that reads as *drug → side effect → required monitoring* is primarily a causal chain (B) but the monitoring atom also reads as a conclusion drawn from the side-effect premise (C): record this as primary B + secondary C. The secondary slot is reserved for genuinely hybrid cases, not routine close-calls.

Pattern notes. When the choice is borderline, when a secondary code is set, or when none of the six codes fit cleanly, the annotator adds a brief pattern note (1–2 sentences) explaining the rationale. Genuine non-fits (truly outside A–F) are flagged for lead-researcher adjudication rather than forced into an ill-fitting code.

E.5 Snippet Text

The snippet text should be a **self-contained verifiable statement**: a reader should be able to judge its correctness without reading the original query. An LLM-drafted initial version is provided; the annotator reviews and edits, resolves pronouns using the shared context (e.g., "the patient" → "a 23-year-old male"), keeps standalone facts as-is, and does not introduce medical facts not present in the source atoms.

E.6 Dual Labels

Each snippet receives **two binary labels**. The notion of "context" refers to the information in the user **query** plus the **earlier statements in the same response**; it does *not* include outside medical knowledge or other entries.

The **in-general** label asks whether the snippet is correct as a general medical statement. For example, "Initiation of clozapine requires a 2-week trial" may be labeled **True** in general if a short initial trial is medically reasonable. The **with-context** label asks whether the same snippet is correct for the specific patient or query. In the treatment-resistant schizophrenia vignette, the same snippet may be labeled **False** with context because the patient would require a longer trial.

When the two labels differ. This is the most research-interesting case: *"generally true, contextually wrong"* (or vice versa). The annotator always adds a note explaining why.

Label values. **True** (medically/factually correct), **False** (factual error), or **Ambiguous** (cannot determine; note required).

E.7 Annotation Dashboard

Annotators interact with the dataset through a browser-based dashboard implementing the workflow described above. Figure 21 shows the main annotation view. The left panel lists the entry's atomic claims with expert true/false labels, plus the original query and the full LLM response (collapsed by default). The right panel holds the per-snippet annotation: shared-context key-value pairs at the top, then one card per snippet showing the selected atom indices, the merged snippet text (LLM-drafted, annotator-edited), the two binary labels (in-general, with-context), an ambiguity flag, a free-text notes field, and the pattern code dropdown (A–F). An "unassigned" counter tracks atoms not yet placed in any snippet, and the entry cannot be marked complete until the counter is zero.

The dashboard supports the workflow steps in §E.2 (define shared context, select atoms, create snippet, generate or edit text, set dual labels, mark ambiguity, add notes, tag pattern) and exports completed annotations as JSON. Two optional features assist annotators without biasing them: a "show suggestions" overlay that visualizes a pre-clustered LLM grouping with purple borders on atoms (off by default), and an "import suggestions" button that loads the LLM groupings as editable snippet cards (the annotator still confirms boundaries and sets all labels).