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

Hasan Iqbal[1] **Sarfraz Ahmad**[1] **Hyunjae Kim**[2] **Sihyeon Park**[3]
Junjie Liao[4,5] **Qingyu Chen**[2] **Preslav Nakov**[1] **Yuxia Wang**[5]
[1]Mohamed bin Zayed University of Artificial Intelligence, [2]Yale University, [3]Korea University
[4]Beijing Normal University, [5]INSAIT, Sofia University, "St. Kliment Ohridski"
hasan.iqbal@mbzuai.ac.ae, yuxia.wang@insait.ai

Project MEDSNIP Code

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 atom-level decomposition can fragment these dependencies, leaving the verifier with clinically incomplete claims. 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 preserves or improves false-class F1, with gains concentrated where answers are long enough to fragment and where the verifier is strong enough to exploit the recovered structure. The largest merge-pattern gain is on causal-conditional clinical chains. It also reduces verifier calls by 24–73%, though the saving survives end-to-end only when decomposition is cheap, which an open-weight decomposer makes possible at no loss of chunking fidelity.

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; Liu et al., 2025; Wang and Zhang, 2024). A fluent but unsupported recommendation can influence patients' decisions about care, mislead downstream systems, and propagate through 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 generation into minimal standalone *atomic* claims, verify each claim independently, and aggregate the verdicts (Chern et al., 2025; 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. However, 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 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.

This motivates a coarser verification unit that preserves these dependencies. We call this unit a *snippet*, a clinically meaningful group of related clauses that is smaller than a full answer but coarser than an atom. It groups the information needed to verify a clinical judgment, such as a finding and its interpretation, a lab value and its reference range, or a diagnosis and 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.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 11081–11104
October 24-29, 2026 ©2026 Association for Computational Linguistics

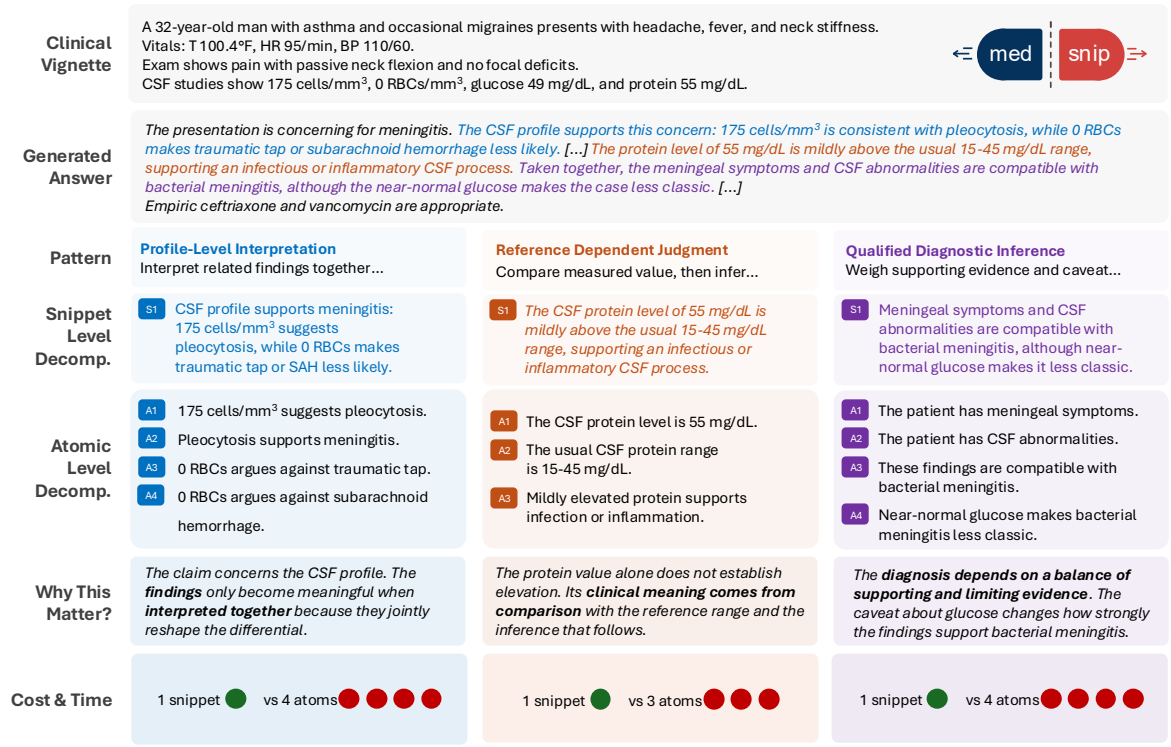


Figure 1: Motivating example from MEDSNIP-BENCH showing how atomization can split clinically coherent judgments. Each column presents one judgment from a clinical-vignette answer. Snippet-level decomposition preserves the linked evidence, interpretation, and caveat as one verifiable unit, while atom-level 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 rather than clinically structured units (Vladika et al., 2024; Pandit et al., 2025). Domain-tailored decomposers and verifier diagnostics similarly remain within the atom-level paradigm (Huang et al., 2026; He et al., 2026). 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 grouped and checked as clinically coherent units that preserve relevant context.
- 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 snippets 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 paired bootstrap confidence intervals throughout, 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., 2025; Wei et al., 2024; Gao et al., 2023; Iqbal et al., 2024; Zerong et al., 2025; Wang et al., 2025, 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.

Motivated by these findings, we take a simpler approach and show that a coarser, clause-grouped unit is sufficient, cheaper, and better 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 abstracts (Vladika et al., 2024), while 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 produces more valid claims than an atomizer (Huang et al., 2026). MEDFACT identifies an over-criticism failure mode in which stronger reasoning verifiers incorrectly mark correct medical text as false (He et al., 2026). Kim et al. (2025) show that retrieval covers only a minority of clinician-identified must-have information and that retrieval-augmented generation (RAG) can degrade factuality. Gunjal and Durrett (2024) make atomic claims interpretable through decontextualization, which is complementary to grouping rather than an alternative to it.

Rewriting "*A because B*" as separate claims can miss errors when each part is plausible but their link is not. Existing methods improve claim-level medical fact-checking, but lack snippet structure and dual in-general and in-patient-context labels. We introduce MEDSNIP-BENCH, which, to the best of our knowledge, is the first human-annotated snippet-level medical fact-checking dataset to preserve clinical claims and their context. Our results support snippet-level verification rather than relying only 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 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 bimodal in query length, so the partition is unambiguous (Appendix A.1).

The atom-level binary factuality labels (i.e., true or false) are obtained from physician annotations inherited from Kim et al. (2025), while MEDSNIP-BENCH then groups these atoms into clinically meaningful snippets and provides the corresponding snippet-level annotations. Every snippet boundary, pattern code and correctness label in MEDSNIP-BENCH was assigned manually.

Snippets and pattern taxonomy: The atomic-claim layer is often too fine-grained for medical content, because clinical judgments can depend on 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 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).

Dataset	Subset / Split	Entries	Units		%F in-general	%F in-context	Labels	Patterns					
			Count	Type				A	B	C	D	E	F
MEDSNIP-BENCH Statistics													
MEDSNIP-BENCH	Parent corpus	276	5,755	atom	–	–	–	–	–	–	–	–	–
	Consumer	140	1,091	snippet	12.4	9.2	dual	634	101	79	194	10	73
	Vignette	136	1,433	snippet	14.6	11.4	dual	374	124	242	423	29	241
	Train	180	1,599	snippet	13.6	10.5	dual	640	143	209	385	26	196
	Dev	51	494	snippet	13.8	10.7	dual	187	44	58	126	10	69
	Test	45	431	snippet	13.7	10.0	dual	181	38	54	106	3	49
	Total	**276**	**2,524**	**snippet**	**13.6**	**10.5**	**dual**	**1008**	**225**	**321**	**617**	**39**	**314**
External Datasets													
HEALTHFC	Atoms	750	1,076	atom	–	–	–	–	–	–	–	–	–
	Snippets (Mode 1)	750	821	snippet	–	–	single	–	–	–	–	–	–
	Snippets (Mode 2)	750	761	snippet	–	–	single	–	–	–	–	–	–
MEDHALLU	Atoms	2,000	3,798	atom	–	–	–	–	–	–	–	–	–
	Snippets (Mode 1)	2,000	2,636	snippet	–	–	single	–	–	–	–	–	–
	Snippets (Mode 2)	2,000	2,586	snippet	–	–	single	–	–	–	–	–	–

Table 1: **MEDSNIP-BENCH and external dataset statistics.** The parent corpus is shown for reference. **%F in-general** and **%F in-context** are false-label rates, and A–F are structural pattern counts. For external datasets, both MEDSNIP modes and their shared atom baseline are reported (Section 4.1). HEALTHFC counts follow question-to-proposition conversion. Dashes indicate unavailable or inapplicable fields.

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 recommendations that are reasonable in general medicine but inappropriate for the vignette at hand.

This distinction appears in practice in MEDSNIP-BENCH, where 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 details and examples are in Appendix B.

3.1 Annotation

Six annotators with biomedical-NLP backgrounds segmented and labeled the data using a shared interface and written guidelines (see Appendix B). Snippet boundaries and pattern codes were based on linguistic and logical judgments over the answer text, rather than clinical judgments requiring medical expertise. Annotators also saw the parent corpus's physician true-or-false labels for each atomic claim, anchoring the task in expert annotations. For each entry, an annotator first identified the shared context (Appendix A.3), such as patient demographics or query topic. They then applied the A–F pattern taxonomy to decide which claims should be merged into a snippet and which should remain separate.

After grouping, the annotator revised each snippet into a self-contained statement and assigned two binary correctness labels. On a stratified inter-annotator agreement (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, following Landis and Koch (1977). Snippet boundary agreement is strong, with a mean pairwise adjusted Rand index (ARI) of 0.723. (see Appendix A.4).

3.2 Dataset Statistics

After annotation, the parent corpus's 5,755 atomic claims map to 2,524 snippets. Of these, 1,554 are multi-atom and 970 are single-atom, with a median of 2 atoms per snippet. Because the dataset skews toward true snippets, we use F_1^F, defined as F1 on false claims, as the primary metric. The train, dev, and test partitions use 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 zero-shot snippet-vs-atom comparison. The retrieval-augmented verifier is developed on dev and confirmed on test. Consumer-health entries favor enumeration, while clinical vignettes contain more reasoning chains, diagnostic conclusions, and standalone 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.

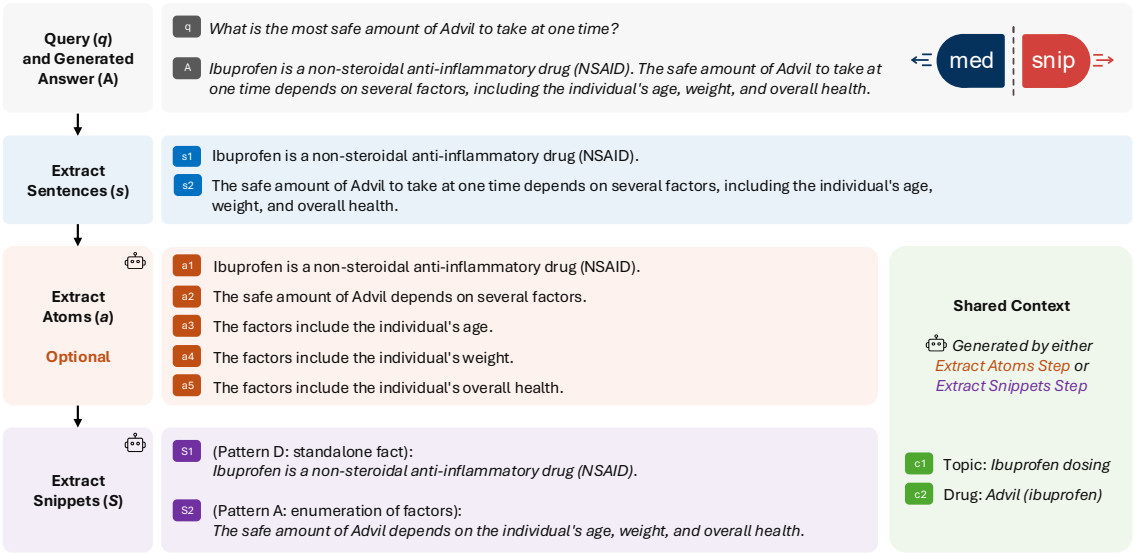


Figure 2: MEDSNIP snippet-generation pipeline on a worked Advil example. After deterministic sentence splitting, Mode 1 extracts atoms and shared context, then clusters atoms into snippets. Mode 2 generates snippets directly from sentences. Both modes return snippets with source pointers, A–F pattern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

From HEALTHFC, we use 327 claims with binary support labels and exclude 423 insufficient-evidence claims from F1 computation. From MEDHALLU, we use 2,000 binary items comprising 1,000 paired ground-truth and hallucinated answers derived from PUBMEDQA (Jin et al., 2019). Table 1 summarizes corpus statistics, splits, structural patterns, and external datasets. Since the external datasets lack snippet boundaries, we apply MEDSNIP to generate snippets before evaluation. Atoms and snippets come from the same decomposition call, so both use identical input and differ only in verification unit. Because HEALTHFC presents claims as questions, we convert them to propositions before decomposition.

4 Method

We compare medical fact-checking pipelines along three axes: *(i)* verification unit, *(ii)* verification context, and *(iii)* verifier design. The main comparison isolates verification granularity by evaluating atoms and snippets. Given a medical answer A, a decomposer produces units $\{u_1, \ldots, u_n\}$, and a verifier labels each as TRUE or FALSE. The atom condition follows the FACTSCORE lineage and verifies the smallest standalone factual statements, while the snippet condition verifies grouped clauses. For MEDSNIP-BENCH, snippets are human-annotated. For HEALTHFC and MEDHALLU, they are generated using the MEDSNIP pipeline.

4.1 MEDSNIP Pipeline

MEDSNIP pipeline converts a long-form answer A and optional query q into snippets S, grouping the answer's own clauses under the question rather than any retrieved evidence. 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. Sentence indices give the decomposer fixed source pointers, which preserve provenance, let us check that every sentence is assigned, and provide the shared index space in which automatic and human snippets are compared. The splitter is rule-based rather than learned, so it introduces no additional model into the pipeline.

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 illustrates both modes using an Advil example, where listed factors are either extracted as atoms and grouped or grouped directly into snippets. Atomize-then-group uses two LLM calls and mirrors human annotation, while snippet-direct uses one. We use GPT-5.4 (OpenAI, 2026) as the decomposer across modes and datasets to avoid confounding the baselines. Section 5.5 varies the decomposer across all six models. Full prompts appear in Appendix C. On the full MEDSNIP-BENCH, atomize-then-group better matches human snippets than snippet-direct in sentence-level F1 (0.756 vs. 0.734) and mean embedding similarity (0.824 vs. 0.794). However, snippet-direct achieves higher F_1^F across all three datasets (Section 5.5). Boundary agreement and verification utility are therefore distinct, so we report both modes. Appendix A.5 provides the full pipeline evaluation. Pattern codes are retained for analysis, while the main per-pattern results use human annotations from 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 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:

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

The verifier reads the accumulated evidence and predicts a label with a confidence score $c_t \in [0, 1]$ emitted with its verdict rather than by a separate calibrator:

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

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 5.6 reports the resulting coverage-F_1^F trade-off. Full Prompts are provided in Appendix C.

5 Experiments and Results

We evaluate whether snippet-level verification improves factuality 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 1, 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).

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

Model	MEDSNIP-BENCH				HEALTHFC			MEDHALLU		
	Atom	Snippet	Mode 1	Mode 2	Atom	Mode 1	Mode 2	Atom	Mode 1	Mode 2
GPT-5.4-high	0.321	**0.431**	**0.429**	**0.418**	0.672	0.672	**0.693**	0.694	**0.720**	**0.721**
GPT-4o	0.344	**0.401**	0.301	0.317	0.689	0.710	**0.736**	0.559	0.568	**0.579**
GEMMA-4-31B-IT	0.340	**0.389**	0.350	0.377	0.664	0.682	0.690	0.612	**0.639**	**0.648**
GPT-OSS-20B	0.344	**0.401**	0.330	0.366	0.680	0.678	0.689	0.623	0.627	0.636
LLAMA-3.3-70B	0.316	0.331	**0.235**	0.280	0.678	0.700	0.693	0.507	**0.482**	0.487
LLAMA-3.1-8B	0.238	0.278	0.201	0.212	0.612	0.598	0.610	0.471	**0.533**	**0.556**

Table 2: F_1^F for CLAIM-ONLY verification across six models and three datasets. Green and red mark gains and losses over atoms, with significant differences in bold (95% confidence interval). MEDSNIP-BENCH uses human snippets, while both pipeline modes use drop-mixed projection. External units share a decomposition call.

The open-weight models are GEMMA-4-31B-IT, 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 prompts, with atom and snippet decompositions produced by GPT-5.4 to isolate verification granularity. For HEALTHFC and MEDHALLU, OR-FALSE predicts an answer as FALSE if any unit is predicted FALSE. MEDSNIP-BENCH requires no answer-level aggregation because labels are at the snippet level. Automatic snippets use drop-mixed projection from human atom labels. Significance uses 95% bootstrap intervals over 10,000 paired entry- or claim-level resamples (see Appendix D).

5.2 Snippet Verification Improves F_1^F

Table 2 compares atoms and snippets across six verifiers and three datasets. On MEDSNIP-BENCH, human snippets outperform atoms for all verifiers, significantly for the four strongest. With automatic snippets, only GPT-5.4-high gains significantly, scoring 0.429 versus 0.431 on human snippets. Four other verifiers fall below their atom baselines, significantly for LLAMA-3.3-70B. Thus, the human-automatic gap is verifier-dependent. Snippet-direct outperforms atomize-then-group for all but the strongest verifier and on the external datasets. It yields two significant HEALTHFC gains and four of eight significant MEDHALLU gains, while atomize-then-group yields none on HEALTHFC. Gains grow with verifier strength and source structure (Section 5.3). Snippets also reduce verifier calls. On MEDSNIP-BENCH, human snippets reduce units by 56%, while atomize-then-group and snippet-direct reduce them by 73% and 63%. The two modes reduce units by 24% and 29% on HEALTHFC, and 31% and 32% on MEDHALLU. Since snippets are longer, MEDSNIP-BENCH verification cost falls by 54%, versus 73% for units. Appendix D.5 reports end-to-end costs.

Corpus	Words	Sent.	Merge	Δ F_1^F
MEDSNIP-BENCH	244	11	3.37	+0.114
MEDHALLU	29	1	1.44	+0.026
HEALTHFC	11	1	1.31	+0.000

Table 3: Source length and grouping opportunity per corpus, with the GPT-5.4 snippet–atom gap. All three columns use atomize-then-group, so the comparison is matched.

Result analysis: The cross-model results support the main granularity claim, with a boundary condition. Snippet-level verification reduces calls by grouping units and improves F_1^F across verifier families, but the improvement is not uniform. It is largest where the source answer is long enough for atomization to fragment a clinical judgement and where the verifier can reason over the recovered structure. Where either condition is absent the effect narrows and can reverse, significantly so for LLAMA-3.3-70B on both MEDSNIP-BENCH and MEDHALLU. 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, see Appendix D for more details.

5.3 Structure and Snippet Gains

Snippet-level verification can only help when there is meaningful structure for atomization to fragment. The three corpora differ substantially in this respect, providing a natural explanation for the variation in gains across datasets.

Pattern	n	Snip	Atom	Δ
A enumeration	1,008	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% bootstrap confidence intervals.

Decomposer	Sent F1	Merge	Δ F_1^F	Cost
GPT-5.4 (high)	**0.756**	3.37	+0.099	$23.63
GEMMA-4-31B-IT	**0.756**	2.08	+0.061	$0.29
LLAMA-3.3-70B	0.742	1.98	+0.095	$0.21
GPT-4O	0.722	2.31	+0.092	$5.79
GPT-OSS-20B	0.706	2.59	+0.055	$0.20
LLAMA-3.1-8B	0.695	1.37	+0.045	$0.03

Table 5: Decomposer quality and downstream effect on MEDSNIP-BENCH, with GPT-5.4 verifying. Every gap is measured against the same expert-atom baseline and every one excludes zero.

Table 3 relates source length and merge ratio to the snippet-to-atom gap under GPT-5.4. MEDSNIP-BENCH has the most structure, with medians of 244 words, 11 sentences, and 3.37 atoms per snippet, and the largest gain of $+0.114$ F_1^F. In contrast, 98% of HEALTHFC claims are single sentences, yielding little structure and no gain. MEDHALLU lies between both datasets in structure and improvement. This ordering holds only with a strong verifier. Weaker open-weight verifiers can lose accuracy on richer MEDSNIP-BENCH snippets (Table 2). Snippet-level verification therefore requires useful source structure and a verifier capable of using it. With three corpora, we report this ordering without fitting a formal relationship.

5.4 Per-Pattern Analysis

The MEDSNIP-BENCH annotations show where snippets help. Table 4 reports pattern-level F_1^F using GPT-5.4 with CLAIM-ONLY. Snippets outperform atoms on all patterns except the small topic-shift category E. The largest merge-pattern gain is for Pattern B, causal-conditional chains, at $\Delta = +0.083$, suggesting that atomization harms verification of clause relations. Pattern E favors atoms, but contains only 39 snippets and its confidence interval crosses zero. For single-atom patterns D and F, boundaries often coincide, so gains may reflect cleaner snippet text rather than granularity. Patterns A, B, and C provide a clearer test. Results also depend on the setting. With FULL-CONTEXT or weaker verifiers, several gaps narrow or reverse, including Pattern B under FULL-CONTEXT with GPT-4O (Appendix D).

Result analysis: The results clarify why snippets help. The largest merge-pattern gain occurs on causal-conditional chains, where a conclusion depends on 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.

5.5 Robustness of the Decomposer

To test dependence on GPT-5.4, we generate snippets with all six models and verify them with all six verifiers in both pipeline modes. Table 5 reports the GPT-5.4 verifier column, while Appendix D.7 gives all 72 combinations. The table includes boundary fidelity, merge ratio, the snippet-direct gap against expert atoms, and decomposition cost for 276 entries. Boundary fidelity does not follow model scale. On the 275 entries completed by both models, GEMMA-4-31B-IT matches GPT-5.4 at 0.756, with higher precision (0.778 vs. 0.723), while costing $0.29 instead of $23.63. LLAMA-3.3-70B also outranks GPT-4O. Rankings instead follow merge ratio. GPT-5.4 groups most at 3.37 atoms per snippet, increasing recall to 0.898 but lowering precision. LLAMA-3.1-8B groups least at 1.37 and reverses this balance. With GPT-5.4 verification, every decomposer outperforms expert atoms, with all gaps significant under an entry-clustered bootstrap. LLAMA-3.1-8B gains $+0.045$ versus GPT-5.4's $+0.099$, so decomposer variation is smaller than the gain over atoms. Boundary fidelity and downstream utility differ. GEMMA-4-31B-IT ties for the best boundary agreement but has one of the smallest verification gains, echoing Section 4.1.

With only six decomposers, we report both rankings without fitting a correlation. Across the full matrix, verifier capability matters more than decomposer identity. Averaged over both modes, verifier column means span 0.113 F_1^F, while decomposer row means span 0.019, a sixfold difference. This favors deployment because decomposition runs once per answer, while verification runs once per unit.

Wording against grouping: Patterns D and F contain one atom each, so any gain there cannot come from grouping. We isolate the two effects by rewriting each atom with the pipeline's decontextualization rules and no grouping, then verifying the rewritten atom. Decontextualization alone recovers 94% of the gain on the keep-atomic patterns and 26% on the merge patterns, leaving 74% of the merge-pattern gain attributable to grouping. Wording explains least on Pattern A at 21%, then Pattern B, the causal-conditional chains that motivate the method, at 25%. The full three-way comparison is in Appendix D.6.

Cost with decomposition included: Reporting verifier calls alone understates what snippet-level verification costs, since building snippets takes an extra call per answer in atomize-then-group. Charging each pipeline for its own decomposition reverses the headline for the configuration used above. With GPT-5.4 decomposing, snippet verification costs 19.1% more end to end than atom verification, because the second decomposition call outweighs the verifier calls it saves. With GPT-OSS-20B decomposing, the same comparison saves 34.1%. The open-weight result above is therefore not only a robustness check. It is what makes the cost reduction survive honest accounting. Appendix D.5 gives the full breakdown.

5.6 Retrieval-Augmented Verification Provides a Coverage Lever

Table 6 compares the retrieval-augmented verifier with the single-call baseline on MEDSNIP-BENCH dev and test. At near-full coverage, retrieval matches but does not improve over the baseline. Higher confidence thresholds trade coverage for 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%. The robust finding is therefore a coverage-F_1^F trade-off, not a full-coverage improvement. Full-coverage ensembles and subset routers improved dev results but did not replicate on test and are reported as negative results in Appendix E.

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 6: Coverage-F_1^F trade-off on MEDSNIP-BENCH with GPT-5.4. Thresholded results exclude abstained and low-confidence units.

Results analysis: Retrieval augmentation serves a different role from snippets. It does not improve on the single-call baseline at full coverage, but enables confidence-based abstention. Excluding uncertain cases raises F_1^F on dev and test while reducing coverage, which may help when evidence and abstention matter. 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 under capable verifiers while reducing verifier calls. The strongest and most interpretable gains occur when atomization separates causal, conditional, or premise-supported clinical reasoning into isolated claims. This benefit depends on available structure and the verifier's ability to use it. 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 D.

Granularity and text quality: When snippet and atom boundaries coincide, gains may come from clearer wording, such as resolved references, rather than granularity. Evidence for granularity is strongest for merge patterns, including enumerations, causal or conditional chains, and conclusions with premises. Appendix D.6 separates these effects by rewriting atoms without grouping. Wording explains nearly all of the keep-atomic gain but only about a quarter of the merge gain. Because repeated verification of identical text changes F_1^F by 0.013, smaller differences may reflect variation rather than an effect.

Sample size: MEDSNIP-BENCH contains 276 entries and 2,524 snippets, which is enough to separate the largest pattern categories but not the smallest. Pattern E holds 39 snippets, and its negative delta has a confidence interval spanning $[-0.300, +0.095]$, so we draw no conclusion from it. The external evidence is similarly bounded, since HEALTHFC scores only the 327 claims that carry binary support labels. Section 5.3 reports an ordering across three corpora rather than a fitted relationship for the same reason.

Pipeline and retrieval design: External-dataset snippets are generated automatically with a single LLM-based decomposer. Section 5.5 varies the decomposer over all six models, but only on MEDSNIP-BENCH, so the decomposer sweep and the external corpora do not overlap.

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 E.

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. We release the annotation guidelines, data, and code for inspection and reuse. Efficient verification could also scale low-quality medical content or overconfident evaluation. Snippet-level verification should support auditing, not replace 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.

Ethan Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2025. FacTool: Factuality detection in generative AI – a tool augmented framework for multi-task and multi-domain scenarios. In *Proceedings of the 2025 Conference on Language Modeling*, Montreal, Canada.

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.

Anisha Gunjal and Greg Durrett. 2024. Molecular facts: Desiderata for decontextualization in LLM fact verification. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3751–3768, Miami, Florida, USA. Association for Computational Linguistics.

Jiayi He, Yangmin Huang, Qianyun Du, Xiangying Zhou, Zhiyang He, Jiaxue Hu, Xiaodong Tao, and Lixian Lai. 2026. MedFact: Benchmarking the fact-checking capabilities of large language models on Chinese medical texts. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pages 604–652, San Diego, California, USA. Association for Computational Linguistics.

Heyuan Huang, Alexandra DeLucia, Vijay Murari Tiyyala, and Mark Dredze. 2026. MedScore: Generalizable factuality evaluation of open-ended long-form medical answers by domain-adapted claim decomposition and verification. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 14149–14180, San Diego, California, United States. Association for Computational Linguistics.

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.

J. Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Fenglin Liu, Hongjian Zhou, Boyang Gu, Xinyu Zou, Jinfa Huang, Jinge Wu, Yiru Li, Sam S. Chen, Yining Hua, Peilin Zhou, Junling Liu, Chengfeng Mao, Chenyu You, Xian Wu, Yefeng Zheng, Lei Clifton,

Zheng Li, Jiebo Luo, and David A. Clifton. 2025. Application of large language models in medicine. *Nature Reviews Bioengineering*, 3(6):445–464.

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.

Dandan Wang and Shiqing Zhang. 2024. Large language models in medical and healthcare fields: Applications, advances, and challenges. *Artificial intelligence review*, 57(11):299.

Yuxia Wang, Minghan Wang, Hasan Iqbal, Georgi N. Georgiev, Jiahui Geng, Iryna Gurevych, and Preslav Nakov. 2025. OpenFactCheck: Building, benchmarking customized fact-checking systems and evaluating the factuality of claims and LLMs. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11399–11421, Abu Dhabi, UAE. Association for Computational Linguistics.

Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Nenkov Georgiev, Rocktim Jyoti Das, and Preslav Nakov. 2024. Factuality of large language models: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19519–19529, Miami, Florida, USA. Association for Computational Linguistics.

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.

Zhaxi Zerong, Chenxi Li, Xinyi Liu, Ju-hui Chen, and Fei Xia. 2025. A systematic survey of claim verification: Corpora, systems, and case studies. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21452–21474, Suzhou, China. 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 MEDSNIP-BENCH, including how we partitioned the parent corpus, created the train, dev, and test splits, measured inter-annotator agreement, and evaluated how faithfully the MEDSNIP pipeline reproduces 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, with consumer-health queries ranging from 2 to 31 words and clinical vignettes from 112 to 254 words, leaving no queries in the 32–111-word gap (Figure 3a). We use a threshold of 80 words to split the 276 entries into 140 consumer-health and 136 clinical-vignette entries. Any threshold between 32 and 111 yields the same partition.

The two subsets differ in verification demands as well as length. Consumer-health queries are short questions about a drug, dose, interaction, or symptom, with answers typically grounded in standard-of-care knowledge. Clinical vignettes provide a structured patient history and ask for a diagnosis, next test, or treatment, so the answer connects premises whose correctness depends on the patient's specific context rather than general medical facts alone.

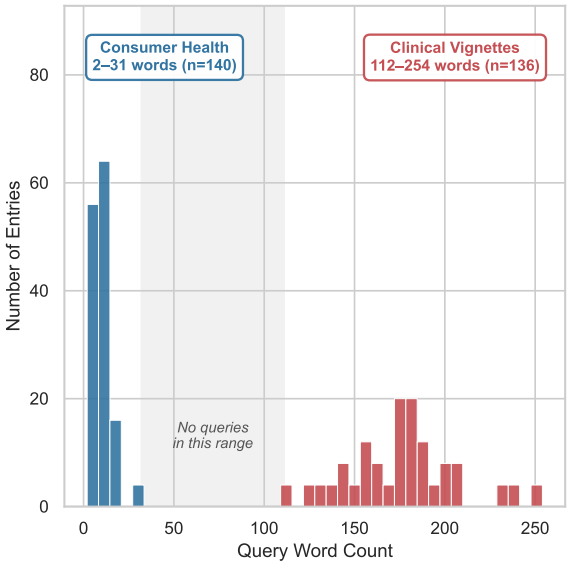
Representative queries: A typical consumer-health query is a single short sentence:

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

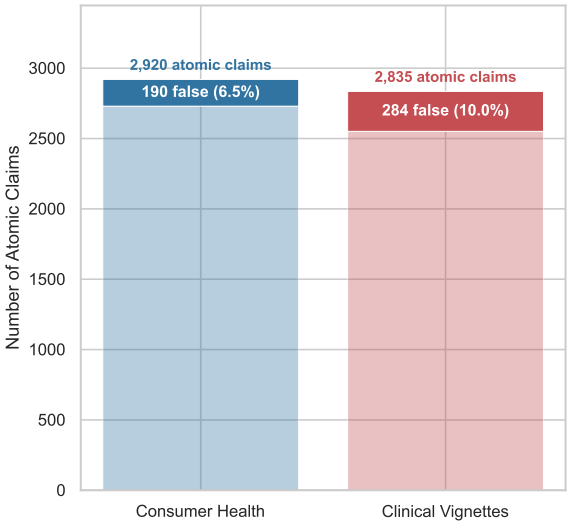
A typical clinical-vignette query is a paragraph-length scenario followed by a question:

> *"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 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.



(a) Query word-length distribution.



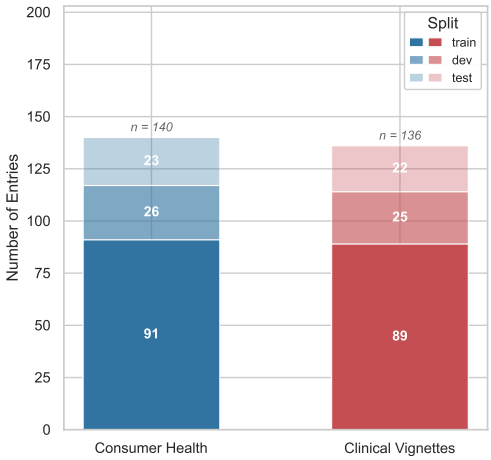
(b) Atomic-claim composition by subset.

Figure 3: Subset selection from the parent corpus.

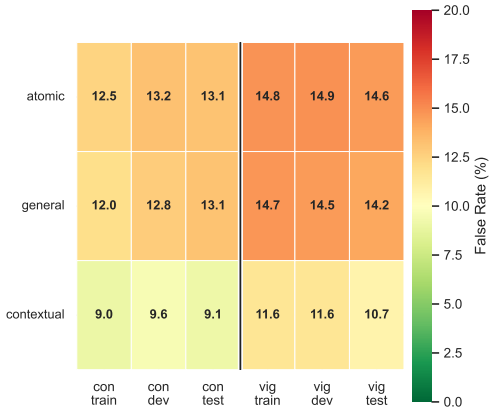
Vignette responses carry a higher atom-level false-rate (10.0%) than consumer responses (6.5%), consistent with the harder clinical-reasoning content (Figure 3b). All downstream subset-conditional analyses preserve this distinction.

A.2 Train / Dev / Test Split

We stratify the 276 entries by source subset and a coarse in-general false-rate bucket, then sweep random seeds to minimize snippet-level false-rate differences across the three label dimensions while maintaining balanced splits. Seed 202 provides the best balance, with a worst-case spread of only 1.1 percentage points across splits. Entry and snippet counts for each split are reported in Table 1.



(a) Per-subset entry counts across splits.



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

Figure 4: Train / dev / test split.

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 snippets in the clinical or topical scenario, supporting 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, allowing annotators to add keys when downstream snippets refer to additional information.

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

Inter-annotator reliability for MEDSNIP-BENCH was computed on a 64-snippet IAA batch annotated independently by all six annotators. Figure 5 shows chunking agreement per entry, Fleiss' κ for each labeled field, and pairwise Cohen's κ for the in-general label.

Binary in-general label: Fleiss' $\kappa = 0.662$ across six annotators (Figure 5b). Pairwise Cohen's κ ranged from 0.385 to 0.947 across 15 annotator pairs, with a mean of 0.663 (Figure 5c). All six annotators agreed on 52 of 64 items (81.3%).

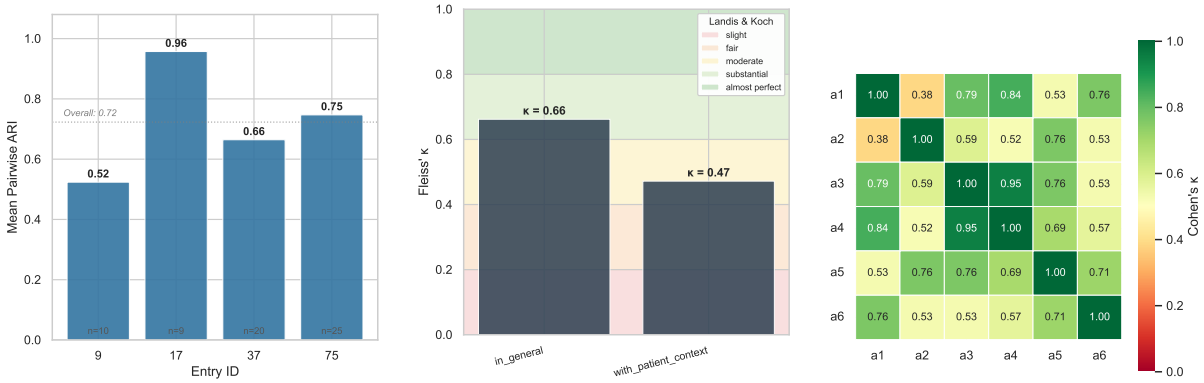
Snippet chunking: Adjusted Rand index (ARI) on snippet partitions gave a mean of 0.723, ranging from 0.524 to 0.957 across annotator pairs (Figure 5a). The more constrained chunking decision therefore shows substantial agreement.

Pattern label: Pattern annotation was integrated into the same annotation pass under the written guidelines (Appendix B). On a held-aside subset annotated by multiple annotators, pattern agreement was $\kappa \approx 0.59$, with the largest residual confusions between A (enumeration) and C (conclusion + premises) for snippets resembling both. The taxonomy and worked anchor cases were re-circulated for calibration before the production pass.

A.5 Auto-Snippet Pipeline Quality

We evaluate the MEDSNIP pipeline against human-annotated MEDSNIP-BENCH across all 276 entries. *Chunking fidelity* measures agreement with human snippet boundaries, while *label robustness* measures sensitivity to the label-projection policy for auto-snippets.

Chunking fidelity: Against human snippet boundaries, Mode 1 (atom-to-snippet) achieves a sentence-set F1 of 0.756, compared with 0.734 for Mode 2 (snippet-direct). Mean embedding cosine similarity shows the same pattern at 0.824 and 0.794, respectively.



(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.

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

Mode 1 therefore more closely reproduces human snippet structure, consistent with its two-pass design that preserves atom-level boundaries as shown in Table 7.

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 labels projected from their source atoms. We compare three policies. *AND* requires all covered atoms to be true, with mixed-label sentences treated as false. *Majority* uses the majority label, with ties resolved as true. *Drop-mixed* removes auto-snippets spanning conflicting human labels and is used throughout the main body. AND and majority serve as robustness checks and agree in sign with drop-mixed in eleven of the twelve verifier-by-mode cells. Projection effects are small relative to model-level differences. Drop-mixed retains 1,382 snippets in Mode 1 and 1,753 in Mode 2, with dropped cases reflecting mixed-label coverage rather than pipeline failures.

B Annotation Guidelines

The guidelines below were distributed to annotators with only operational and identifying details removed. The pattern taxonomy and worked anchor examples are reproduced unchanged.

B.1 Background

The dataset contains 5,755 atomic claims from 276 medical QA entries. Annotators group these atoms into snippets and assign dual labels. Each entry shows:

- **Query**: the medical question.
- **Full Response**: the LLM-generated answer.
- **Atomic Claims**: statements extracted from the response with expert true or false labels.

Why we merge at all: Atomic-claim extraction can be too fine-grained. An atom such as *Option B is the correct answer"* or *The factors include the individual's age"* can be unverifiable without surrounding reasoning. Medical responses, especially clinical vignettes, often connect claims through *because / since / therefore / requires*. Splitting these chains removes context needed for verification.

We therefore merge atoms when separating them would break a logical dependency. Every rule below and each consensus judgment follows the same question:

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

If **yes**, keep it atomic. If **no**, merge it with the atoms that provide the missing context.

Code		Example	Decision and Rationale
		Patterns that trigger MERGE	
Pattern A		**Enumeration of properties of one subject.** Multiple features, factors, or properties describe the same subject. The enumeration is the verifiable unit.	
A.1		*Consumer query on safe Advil dose*	**MERGE.** Atoms 2–4 begin with "The factors include…" and are meaningless without Atom 1.
	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."	
A.2		*Vignette on a 72-year-old man with suspected leukemia*	**MERGE.** Three clinical features that together form the "CLL picture."
	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."	
Pattern B		**Causal or conditional chain.** Atoms are connected by *because*, *since*, *as*, *due to*, *requires*, or *therefore*, or one atom serves as the premise for another.	
B.1		*Vignette on a 23-year-old with treatment-resistant schizophrenia*	**MERGE.** Atom 7 states the condition; Atoms 8 and 9 specify the two required failures.
	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."	
B.2		*Consumer query on Hepatitis A IgM*	**MERGE.** Atom 8 is the reason for Atom 7.
	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."	
Pattern C		**Conclusion + supporting premises.** A recommendation, diagnosis, or final answer is meaningful only when read with the reasoning or referent that supports it.	
C.1		*Same schizophrenia vignette, end of response*	**MERGE.** Atom 20 is unverifiable without Atom 19.
	Atom 19	"The most appropriate next step in management is to initiate clozapine."	
	Atom 20	"Option B is the correct answer."	
C.2		*Same vignette, earlier in response*	**MERGE.** Atoms 1–2 give the differential; Atom 3 is the conclusion drawn from it.
	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."	

Table 8: **Structural patterns that trigger merging.** Patterns A–C capture cases in which atomic decomposition removes context required for verification. The worked examples were used as inter-annotator anchors.

B.2 Task

For each entry, the annotator defines shared context as key-value pairs describing patient demographics, medications, or topic, then groups related atoms into snippets using an editable LLM draft. Each snippet receives two labels, *(i). with context*, indicating correctness for the patient or query, and *(ii). in general*, indicating medical correctness. The annotator assigns a structural pattern (A–F), marks ambiguity when correctness cannot be determined, and adds notes when labels differ.

B.3 Shared Context

Shared context is defined once per every entry and applies to all snippets, capturing information from the original query that may be needed to interpret and verify individual statements. For clinical vignettes, typical keys include *age*, *sex*, *chief_complaint*, *medical_history*, *medications*, *vitals*, *treatment_history*, and *correct_answer*. For consumer-health queries, where less patient-specific information is required, typical keys include *topic*, *drug_a*, and *drug_b*.

Code		Example	Decision and Rationale
Patterns that justify KEEPING ATOMIC			
Pattern D		**Complete standalone fact, definition, or lab interpretation.** The atom is a self-contained medical statement and requires no surrounding text for verification.	
D.1		*Hepatitis A IgM definition*	**ATOMIC.** A complete definition; nothing else is needed.
	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."	
D.2		*Lab value interpretation*	**ATOMIC.** A single lab-value claim verifiable in isolation.
	Atom 1	"A platelet count of 119,000/mm^3 is within normal limits." *This claim is false (normal range: 150,000–400,000), but factuality is a labeling question rather than a grouping decision.*	
Pattern E		**Topic genuinely shifts.** The response moves from one subject to an unrelated one.	
E		*Example* • The response may shift from *drug composition to overdose risk*, or from one differential diagnosis to a different one. • The claims concern substantively different subjects and do not depend on one another for verification.	**ATOMIC.** Split at the topic boundary.
Pattern F		**Same topic, but each atom carries a different checkable fact.** Nearby atoms concern the same subject but each expresses an independently verifiable fact.	
F		*Example* • Two claims about the same drug can remain separate if they convey distinct information. • A standalone fact does not need to be merged into a nearby enumeration merely because it appears next to one. • The "isolated [false] atom" idiom also falls under Pattern F: a false atom is kept separate so that it does not contaminate adjacent true claims.	**ATOMIC.** Keep independently verifiable facts separate.

Table 9: **Structural patterns that justify keeping atoms separate.** Patterns D–F preserve claims that remain independently verifiable despite appearing near related content.

B.4 Grouping Rules

Tables 8 and 9 give the six structural patterns used to operationalize this decision, together with the worked examples used as inter-annotator anchors. 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. 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.

Hybrid snippets: A snippet may carry an optional secondary code when two patterns clearly apply. For example, a snippet describing a drug, its side effect, and the resulting need for monitoring is primarily a causal chain (B), while the monitoring atom also functions as a conclusion supported by the side-effect premise (C).

This is recorded as primary B and secondary C. The secondary code is reserved for genuinely hybrid cases rather than routine close calls. When the choice is borderline, when a secondary code is set, or when none of the six codes fit cleanly, the annotator adds a 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.

B.5 Snippet Text

The snippet text should be a self-contained, verifiable statement that can be judged without reading the original query. An LLM-generated draft is provided for the annotator to review and edit. The annotator resolves pronouns using the shared context, replaces references such as the patient" with a 23-year-old male," keeps standalone facts unchanged, and does not introduce medical information absent from the source atoms.

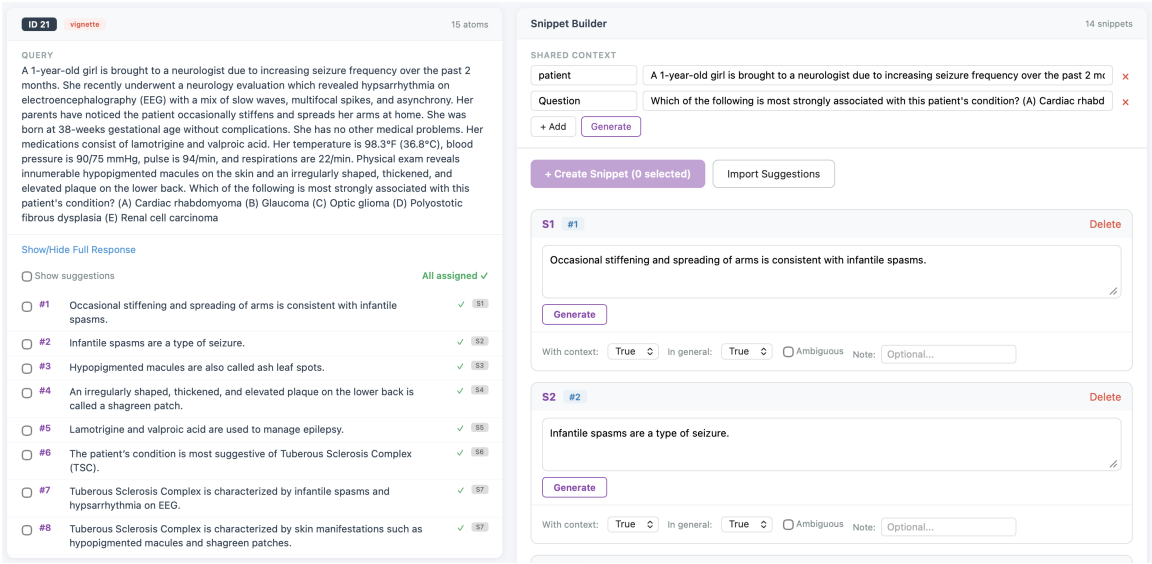


Figure 6: **Annotation dashboard.** The left panel shows atomic claims with expert labels, the query, and the full LLM response. The right panel shows shared context and per-snippet cards containing atom selection, merged text, dual labels, ambiguity status, notes, and pattern code. All annotator-identifying elements have been removed.

B.6 Dual Labels

Each snippet receives two labels. Here, context" refers to information in the user query and earlier statements in the same response, excluding external medical knowledge and other entries. The *in-general* label indicates 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 indicates whether the same snippet is correct for the specific patient or query. In a treatment-resistant schizophrenia vignette, the same statement may be labeled FALSE with context if the patient requires a longer trial. When the two labels differ, the statement is generally correct but contextually wrong, or vice versa, and the annotator adds a note explaining the difference. Labels are **True** for medically or factually correct statements, **False** for factual errors, and **Ambiguous** when correctness cannot be determined. Ambiguous labels also require a note.

B.7 Annotation Dashboard

Annotators use a browser-based dashboard implementing the workflow in Appendix B.2. As shown in Figure 6, the interface presents the query, response, and atomic claims with expert labels, alongside controls for grouping atoms into snippets, editing snippet text, assigning dual labels and pattern codes, and recording ambiguity or notes.

The dashboard ensures that all atoms are assigned before an entry is completed and exports annotations as JSON.

C 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.

C.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 7 and 8 show both prompts verbatim.

C.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 10. 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 9–13.

Component	Mode 1 Extract	Mode 1 Cluster	Mode 2 Direct
Merge patterns	–	✓	✓
Atom extraction	✓	–	–
Granularity rules	✓	–	–
Context schema	✓	✓	✓
Snippet text rules	–	✓	✓
Output JSON	`extract`	`cluster`	`direct`

Table 10: **Composition of reusable prompt components.** Mode 1 separates atom extraction from clustering, whereas Mode 2 constructs snippets directly. Checkmarks indicate the components included at each stage.

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 7: 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 8: Baseline verifier in FULL-CONTEXT mode. The full answer is for disambiguation only.

Task Heads and Output Schemas

Each prompt ends with a mode-specific task and JSON schema (Figures 14–16).

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 9: 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 10: 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 11: Per-subset snippet wording rules. Used by Mode 1 cluster and Mode 2.

C.3 Retrieval-Augmented Verifier

The retrieval verifier (Section 4.3) follows the prompt sequence in Figures 17–20 during iterative verification. If no verdict is reached, the force-final fallback in Figure 20 is used.

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 12: Per-subset atom granularity rules for consumer-health and clinical-vignette entries. Used by Mode1 step1 to guide atomic claim extraction.

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 13: Atom-definition and sentence-mapping rules for extracting self-contained atomic claims. 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 14: Mode 1 step 1 task and output schema for atomic claim extraction. Combines with the extraction-common block, granularity rules, and the per-subset context schema.

Task head: Mode 1 step 2 (cluster)

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 15: 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)

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 16: 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 17: 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 18: 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 19: Verifier truth-judgement rules for general medical statements, including compound-claim materiality and handling of mixed or inconclusive evidence.

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 20: Force-final fallback prompt used at the iteration cap when no verdict is reached. Biased toward abstention under residual uncertainty.

D Experimental Details

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

D.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 11.

Open-weight models: GEMMA-4-31B-IT, GPT-OSS-20B, LLAMA-3.3-70B, and LLAMA-3.1-8B 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 11.

Retrieval backends: The retrieval-augmented verifier uses Google Serper API ($0.001 per query) and a public PubMed biomedical-literature API (free, abstracts only). Both return $k = 3$ snippets per query. Each verified item issues between one and five retrieval rounds depending on confidence-gated stopping, for an average of ≈ 1.9 retrievals per item.

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

D.2 Bootstrap CI Methodology

Every confidence interval for an F_1^F difference is a 95% bootstrap interval over $B = 10{,}000$ replicates, with significance declared when the interval excludes zero. All resampling is paired, so each replicate recomputes both arms on the same draw and Δ is a paired difference. Two interval types are used depending on the resampling unit. For flat samples, as in the per-pattern comparisons of Table 4, we use bias-corrected and accelerated (BCa) intervals. The bootstrap distribution of Δ can be skewed when either F1 approaches zero, as in small pattern cells or under weak verifiers, and BCa corrects for bias and skewness.

When replicates resample clusters, we instead use percentile intervals. For Table 2 and Table 14, we resample whole source entries on MEDSNIP-BENCH and whole claims on the external corpora. This accounts for dependence among snippets from the same answer and for the different numbers of units produced by the two arms. Because the resulting statistic does not have the flat structure required for the BCa jackknife correction, we report percentile intervals. The paired clustering also tends to narrow the intervals, since errors shared across the two arms cancel within each draw. The bold cells of Table 2 are the 17 of 42 whose interval excludes zero, 7 of 18 on MEDSNIP-BENCH, 2 of 12 on HEALTHFC, and 8 of 12 on MEDHALLU.

D.3 Full Per-Cell Results

Table 11 provides the per-split MEDSNIP-BENCH results underlying the aggregate numbers in Table 2. It reports atom and snippet verifier-call counts, false-class F1, and dollar cost for each evaluated split. Figure 21 visualizes the corresponding cost–F_1^F trade-off, showing how snippet-level verification shifts the baselines toward fewer verifier calls

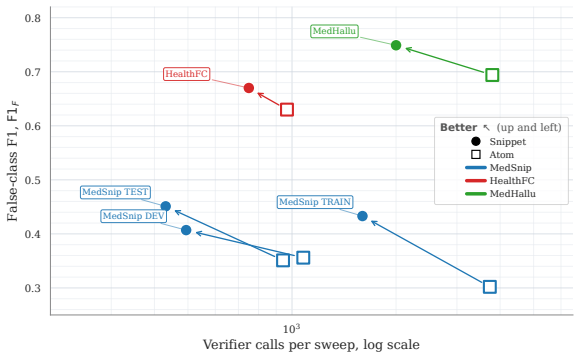


Figure 21: Verifier calls versus F_1^F for snippet- and atom-level baselines across three datasets. Snippets require fewer calls while preserving or improving F_1^F.

D.4 Aggregation Rule Sensitivity

For datasets with answer-level labels, unit-level verdicts must be combined into a single prediction. We test whether this choice affects the snippet–atom comparison using five aggregation rules. OR-FALSE predicts FALSE if any unit is false, following the FACTSCORE default. AND-FALSE requires all units to be false, MAJORITY uses majority vote, THRESHOLD-k predicts FALSE when at least $k = 2$ units are false, and a fifth rule takes the first unit alone.

Across 14 dataset–granularity–mode–model settings, we recompute the snippet–atom F_1^F gap and its 95% paired bootstrap interval under each rule.

Findings: The magnitude of the snippet–atom gap varies with the aggregation rule, but the advantage of snippets holds in most settings. Under THRESHOLD-k, snippets outperform atoms in 12 of 14 settings with $P(\Delta > 0) \geq 0.95$. The two exceptions occur on MEDSNIP-BENCH with GPT-4O in FULL-CONTEXT mode, where predictions are nearly degenerate. Under OR-FALSE, snippets win in 7 of 14 settings. The main exception is AND-FALSE, which reverses the comparison on MEDHALLU by 0.015 F_1^F in favor of atoms ($P = 0.036$). This rule favors atomization because it requires every unit to be FALSE, and atomization produces more units.

Implication: Aggregation is therefore not a neutral implementation choice. It can change both the magnitude and direction of a granularity comparison, so decomposition-based evaluations should state the aggregation rule explicitly. The cost reduction reported in the main paper is unaffected because it depends only on the number of verification units.

D.5 End-to-End Cost

The cost analysis in Section 5.2 considers verification alone, but decomposition also adds overhead. An atom pipeline requires one decomposition call per answer, while atomize-then-group requires a second for grouping. Table 12 includes this cost for MEDSNIP-BENCH with GPT-5.4 as the verifier. With GPT-5.4-high, decomposition costs \$23.63 and makes the full pipeline 19.1% more expensive. With GPT-OSS-20B, it costs only \$0.20, preserving nearly all of the 35.3% verification saving. Snippet-direct further reduces this overhead by requiring only one decomposition call.

Call counts show the same pattern without relying on model prices. Atomize-then-group reduces total calls by 64.7% on MEDSNIP-BENCH but increases them by 14.5% on MEDHALLU and 27.1% on HEALTHFC. Snippet-direct instead yields reductions of 60.1%, 20.9%, and 17.3%, respectively. Decomposition overhead therefore matters most for short answers, where fewer verification calls can be eliminated, consistent with Section 5.3.

Model	Split	Atom calls	Snippet calls	Atom F_1^F	Snippet F_1^F	Δ	Atom cost	Snippet cost
MEDSNIP-BENCH CLAIM-ONLY Verification								
GPT-5.4 (high)	Train	3,735	1,599	0.302	0.433	+0.131	$10.27	$7.13
	Dev	1,078	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	3,735	1,599	0.328	0.416	+0.088	$1.70	$0.80
	Dev	1,078	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	3,735	1,599	0.324	0.394	+0.070	$0.05	$0.02
	Dev	1,078	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	3,735	1,599	0.323	0.400	+0.077	$0.10	$0.06
	Dev	1,078	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	3,735	1,599	0.299	0.322	+0.023	$0.05	$0.03
	Dev	1,078	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	3,735	1,599	0.216	0.287	+0.071	< $0.01	< $0.01
	Dev	1,078	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-BENCH FULL-CONTEXT Verification								
GPT-5.4-high	Train	3,735	1,599	0.475	0.475	−0.000	$17.67	$11.27
	Dev	1,078	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	3,735	1,599	0.338	0.301	−0.037	$7.12	$3.25
	Dev	1,078	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 11: **Per-split verification results on MEDSNIP-BENCH.** Atom and snippet calls give verifier-call counts, with corresponding false-class F_1^F. Δ is the snippet–atom F_1^F difference, with gains and losses shown in green and red. Cost estimates one complete verification sweep.

Decomposer	Decomp. cost	Verification reduction	End-to-end reduction
GPT-5.4-high	$23.63	+54.3%	−19.1%
GPT-4O	$5.79	+33.3%	+8.8%
GEMMA-4-31B-IT	$0.29	+22.6%	+21.3%
LLAMA-3.3-70B	$0.21	+26.0%	+24.8%
GPT-OSS-20B	$0.20	+35.3%	+34.1%
LLAMA-3.1-8B	$0.03	+19.1%	+18.9%

Table 12: End-to-end cost reduction from snippet-level verification across different decomposers, comparing verification cost savings before and after accounting for decomposition overhead.

Pattern	Decision	Raw	Rewrite	Snippet	Gain
A	Merge	0.267	0.292	0.387	21%
B	Merge	0.245	0.279	0.383	25%
C	Merge	0.302	0.354	0.426	42%
D	Keep	0.250	0.336	0.364	75%
F	Keep	0.543	0.624	0.621	103%
A–C	**Merge**	**0.273**	**0.306**	**0.399**	**26%**
D–F	**Keep**	**0.365**	**0.449**	**0.454**	**94%**

Table 13: **Effect of decontextualization by structural pattern.** Raw, rewritten, and snippet columns report F_1^F. Wording gain is the percentage of the raw-to-snippet improvement recovered by rewriting alone.

D.6 Wording Against Grouping

Snippet-level gains can come from grouping dependencies or making snippets easier to verify. Patterns D, E, and F provide a control because they pair one atom with one snippet, so gains must come from wording. To separate these effects, we rewrite each atom using decontextualization rules without grouping, then compare the raw atom, rewritten atom, and snippet.

Rewriting explains 94% of the gain for keep-atomic patterns but only 26% for merge patterns, leaving the remaining 74% attributable to grouping related atoms. Pattern B shows the clearest effect, with wording explaining only 25% of its gain, consistent with its causal and conditional structure where claims depend on one another. Pattern E has only 39 units and negative deltas throughout, so we do not interpret it further.

Mode 1: Atomize-then-Group						
Decomposer ↓	**Verifier →**					
	GPT-5.4	**GPT-4o**	**GEMMA-4-31B-IT**	**GPT-OSS-20B**	**LLAMA-3.3-70B**	**LLAMA-3.1-8B**
GPT-5.4-high	**+0.114**	−0.038	+0.014	−0.009	**−0.077**	−0.035
GPT-4o	**+0.084**	−0.038	−0.005	+0.024	−0.045	+0.015
GEMMA-4-31B-IT	**+0.055**	−0.002	−0.025	−0.015	−0.044	−0.004
GPT-OSS-20B	**+0.066**	−0.036	−0.026	−0.001	**−0.068**	−0.044
LLAMA-3.3-70B	**+0.051**	−0.044	−0.021	−0.013	**−0.063**	−0.032
LLAMA-3.1-8B	+0.032	−0.008	+0.017	−0.018	−0.004	+0.036

Mode 2: Snippet-Direct						
Decomposer ↓	**Verifier →**					
	GPT-5.4	**GPT-4o**	**GEMMA-4-31B-IT**	**GPT-OSS-20B**	**LLAMA-3.3-70B**	**LLAMA-3.1-8B**
GPT-5.4-high	**+0.099**	−0.027	+0.038	+0.022	−0.037	−0.025
GPT-4o	**+0.092**	−0.032	−0.005	+0.033	−0.029	+0.031
GEMMA-4-31B-IT	**+0.061**	+0.005	−0.000	−0.002	−0.023	+0.035
GPT-OSS-20B	**+0.055**	+0.005	+0.002	−0.025	−0.033	+0.030
LLAMA-3.3-70B	**+0.095**	−0.050	+0.001	+0.001	**−0.059**	+0.037
LLAMA-3.1-8B	**+0.045**	−0.018	+0.012	+0.025	−0.018	+0.011

Table 14: **Snippet–atom F_1^F gap across decomposer–verifier pairs on MEDSNIP-BENCH.** Rows denote decomposers and columns denote verifiers. Green cells indicate gains and red cells indicate losses. Bold values have 95% entry-clustered bootstrap confidence intervals excluding zero, based on 10,000 resamples.

There are two caveats. For 382 of 4,868 merge atoms, the rewriter used information from the shared context to effectively reconstruct the merge. We exclude these cases because they do not provide a valid wording-only ablation. In addition, repeating verification on identical text produces a baseline difference of 0.013 F_1^F, so smaller differences are not informative.

D.7 Full Decomposer-by-Verifier Matrix

Section 5.5 reports a subset of the full decomposer-by-verifier experiment. Table 14 shows all six decomposers paired with all six verifiers, under both pipeline modes on MEDSNIP-BENCH. Each cell reports the difference between snippet and expert-atom F_1^F for the same verifier. Because MEDSNIP-BENCH inherits its atomic claims from Kim et al. (2025), all decomposers are compared against the same human-annotated atom baseline. The results show that the verifier matters more than the decomposer. All significant gains occur with GPT-5.4, while all significant losses occur with LLAMA-3.3-70B. No other verifier produces a significant difference. With GPT-5.4, the six decomposers yield gains from +0.032 to +0.114, with all but one reaching significance. Even the 31B and 8B open-weight decomposers therefore produce snippets that outperform expert atoms.

Across verifiers, however, this advantage disappears, suggesting that gains depend mainly on the verifier's ability to use the recovered structure rather than the model used to recover it. Mode 1 and Mode 2 perform similarly overall, supporting our decision to report both rather than select a default.

E 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 performance on the development split but did not provide a stable or consistent improvement on the held-out test split. We therefore report the simpler confidence-thresholded verifier in the main paper and treat these full-coverage routing and ensembling strategies as future work rather than part of the core result.